7

O pequeno programa a que chamamos «rotina» é uma ferramenta muito útil para efectuar funções determinadas no programa em que for inserido, especialmente se tiver de ser utilizado repetidas vezes.

Dispondo deste repositório de rotinas, o programador vê a sua tarefa muito facilitada, pois fica desobrigado de resolver inúmeros problemas que dão muito trabalho e pouco prazer – por exemplo, ordenar números, validar entradas de dados, arredondar números não inteiros, efectuar análises estatísticas, operar com matrizes, imprimir em tamanho duplo e muitas coisas mais.

Esta obra tem ainda uma função pedagógica, mostrando como se elaboram programas eficazes e sem «parasitas».

Todos os programas deste livro foram verificados e testados pelo Gabinete Verbo de Informática.

BIBLIOTECA VERBO DE INFORMÁTICA

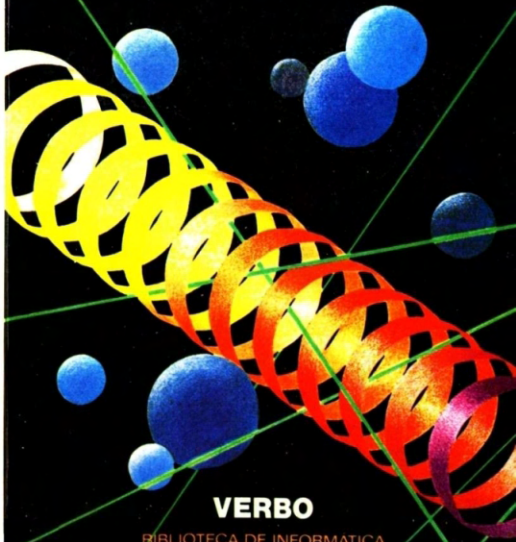
57 ROTINAS EM BASIC W. JOHNSON

Verbo

57 ROTINAS EM BASIC

W. JOHNSON

PARA O SPECTRUM



VERBO

BIBLIOTECA DE INFORMÁTICA

57 Rotinas em Basic
para o Spectrum

W. JOHNSON

57 Rotinas em Basic para o Spectrum

Verbo

Título do original inglês:
Fifty Subroutines for the Sinclair Spectrum
Versão portuguesa e revisão científica
do Carlos Sebastião e Silva

Editorial Verbo, Lisboa/São Paulo
N.º Ed. 1652
Composto por Fotocompográfica, Lda.
Impresso por Empresa Litográfica do Sul
em Novembro de 1985
Depósito legal n.º 9559/85

ÍNDICE

Prefácio

Esta colecção de rotinas (ou sub-rotinas; ao longo da obra serão nomeadas indiferentemente por qualquer destas formas) foi compilada para publicação a partir de um conjunto de programas elaborados para o *ZX Spectrum*. É uma colecção de muito interesse para uma vasta gama de utilizadores do *Spectrum*, pois contém rotinas de referências úteis, assim como rotinas que são difíceis de encontrar ou cuja elaboração é maçadora.

Vale a pena estudar pormenorizadamente as rotinas apresentadas, em razão das técnicas utilizadas e das vantagens que podem trazer à elaboração dos nossos próprios programas. Quantas vezes não dissemos (ou pensámos) já: «Não imaginava que se pudesse fazer isso dessa forma», ou «Não me lembro de ter visto isso no manual!»

Parte do atractivo de lidar com computadores, passado o primeiro entusiasmo que os jogos despertam, está em elaborarmos os nossos próprios programas, sejam eles de jogos ou de aplicação útil, e fazermos o computador executar o que desejamos com um mínimo de instruções e de ocupação de espaço de memória. Há sempre grande satisfação em obter uma solução elegante para um problema.

As rotinas deste livro estão escritas no BASIC do *Spectrum*, mas podem ser facilmente traduzidas para outros dialectos de BASIC com um pouco de cuidado e algumas tentativas.

Foram envidados todos os esforços para assegurar que as rotinas funcionem dentro dos limites especificados, e de uma forma eficiente, mas não se oferecem prémios a quem descubra casos em que elas não funcionem!

Para terminar, apresentamos um programa completo no fim do livro, programa para o qual foram concebidas originalmente pelo

menos uma dúzia das rotinas aqui incluídas. Para fazer com que o programa coubesse à vontade num *Spectrum* de 16 k, foram utilizadas algumas rotinas alternativas, e o programa constitui assim um bom exemplo de economização de memória. O programa em questão permite que sejam desenhadas todas as formas básicas de cristais cúbicos, pela simples introdução de três números que representam a forma cristalina apropriada.

Estrutura das rotinas

Cada rotina apresentada é constituída por duas ou três partes. A parte inicial, separada da rotina propriamente dita, ficará normalmente integrada no corpo principal do programa e gera a informação essencial de que a rotina necessita.

Contudo, dá-se uma sub-rotina de *input* simples para permitir que a rotina seja executada como um programa completo ou que seja testada quando for introduzida no computador.

O bloco principal de instruções é a rotina propriamente dita. Apresentamo-la sem números de linha, de modo a poder ser introduzida de acordo com o programa principal (que a vai utilizar). As linhas de destino das instruções de salto (GO TO, GO SUB) são indicadas por meio de letras (que deverão, obviamente, ser substituídas por números de linha quando a rotina for utilizada). Quando uma instrução ou linha de programa ocupa mais do que uma linha de *écran* (32 caracteres), a segunda linha e as linhas subsequentes são apresentadas neste livro ligeiramente recuadas em relação ao início das restantes, para evitar confusões, embora seja geralmente bem claro onde é necessário incluir um número de linha (ou seja, onde começa uma nova linha). É óbvio que este «recuo» não tem cabimento quando a rotina é introduzida no computador. De notar também que a maioria das linhas contém várias instruções (também denominadas «declarações»), de modo a economizar espaço de memória.

Por fim, acrescenta-se uma sub-rotina de *output* ao final da rotina para dar saída *(output)* ao resultado final desta, para a testar quando é introduzida no computador ou, se a rotina é utilizada como programa isolado, para dar ou apresentar o resultado no *écran*.

Neste livro utilizamos a letra R para indicar o RETURN da rotina principal (ou seja, a instrução que faz com que a sequência de execução volte ao programa principal) e também para indicar o início da secção final (a qual pode, ou não, ser omitida).

Deve ter-se o cuidado de, ao utilizar estas rotinas, verificar se as variáveis nelas utilizadas (principalmente i, j, k, p, q, n, t e z no que respeita a variáveis representadas por uma única letra) não coincidem e, portanto, não entram em conflito com as variáveis utilizadas no programa principal.

Cada rotina deste livro é acompanhada por uma breve explicação do seu funcionamento.

Rotinas

1 ORDENAÇÃO ANGULAR

Esta sub-rotina coloca uma lista de coordenadas por ordem angular relativamente ao seu centro.

As primeiras oito linhas localizam o maior e o menor valor de x e de y e determinam o ponto xm,ym de modo a que se situe a igual distância dos pontos cujas coordenadas são os valores determinados. Seguidamente, calcula-se e analisa-se o ângulo formado entre a horizontal e a linha que une este ponto «médio» a cada um dos pontos «originais», de forma a determinar-se o quadrante ao qual ele pertence. Este procedimento deve-se ao facto de a tangente de um ângulo ser positiva no primeiro e terceiro quadrantes e negativa nos outros dois. O ângulo é então corrigido pela adição de PI ou de 2*PI, conforme for necessário, e armazenado em B(n). Para pôr os ângulos em ordem, usa-se uma sub-rotina de ordenação pelo método dito de «bolhas» *(bubble-sort)*, indicada pela letra B na listagem abaixo.

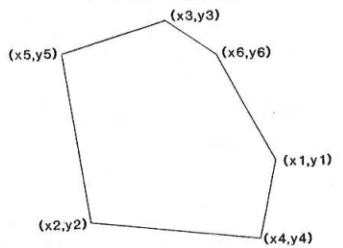
```
INPUT "No. de pares de coorde
 nadas? ";n: DIM A(2,n): FOR
 p=1 TO n: INPUT "x";(p);"=";
 A(1,p),"y";(p);"=";A(2,p)
NEXT p
```

ou utiliza-se a rotina de *input* de dados (17) para determinar n e A(2,n)

```
LET x max=0: LET y max=0
LET x min=A(1,1): LET y min=A
 (2,1)
FOR p=1 TO n
IF A(1,p)>=x max THEN LET x m
 ax=A(1,p)
IF A(2,p)>=y max THEN LET y m
 ax=A(2,p)
IF A(1,p)<x min THEN LET x mi
 n=A(1,p)
IF A(2,p)<y min THEN LET y mi
 n=A(2,p)
NEXT p
LET xm=(x max+x min)/2: LET y
```

```
 m=(y max+y min)/2
DIM B(n): FOR p=1 TO n
LET z=ATN ((A(2,p)-ym)/(A(1,p
 )-xm))
IF A(1,p)>=xm AND A(2,p)>=ym
 THEN LET B(p)=z
IF A(1,p)<xm AND A(2,p)>=ym O
 R A(1,p)<xm AND A(2,p)<ym TH
 EN LET B(p)=PI+z
IF A(1,p)>=xm AND A(2,p)<ym T
 HEN LET B(p)=2*PI+z
NEXT p
B LET q=0: FOR p=1 TO n-1
IF B(p+1)<B(p) THEN GO SUB A:
 LET q=q+1
NEXT p: IF q<>0 THEN GO TO B
GO TO R
A LET z=B(p+1): LET B(p+1)=B(p)
 : LET B(p)=z
LET z=A(1,p+1): LET A(1,p+1)=
 A(1,p): LET A(1,p)=z
LET z=A(2,p+1): LET A(2,p+1)=
 A(2,p): LET A(2,p)=z
RETURN
R RETURN
R PLOT A(1,1),A(2,1): REM IF 0<
 =x min<=255, 0<=y min<=175,
 etc
FOR p=1 TO n-1
DRAW A(1,p+1)-A(1,p),A(2,p+1)
 -A(2,p): NEXT p
DRAW A(1,1)-A(1,n),A(2,1)-A(2
 ,n)
```

Ordenação de ângulos



2 ANUIDADES

Estas sub-rotinas calculam as tabelas de anuidades $A\overline{n}|$ para diferentes taxas de juro anuais e para pagamentos anuais ou mensais.

As operações financeiras baseadas no juro composto têm a série geométrica como alicerce. Assim, sendo i a taxa de juro, 1 milhar de escudos aumentará para $(1+i)^n$ milhares de escudos no espaço de n anos, de modo que, invertendo as coisas, $1/(1+i)^n$ de milhar de escudos tornar-se-á 1 milhar de escudos no espaço de n anos. A anuidade é a soma dos valores $1/(1+i)^n$ vigentes para cada um dos futuros n anos. Deste modo,

$$A\overline{n}| = v+v^2+v^3+\ldots\ldots+v^n$$

onde $v=1/(1+i)$

Ora a soma dos n primeiros termos desta série é igual a $(1-v^n)/1$, logo $A\overline{n}| = (1-v^n)/1$, expressão esta que é utilizada para calcular as tabelas apresentadas pelas duas rotinas listadas abaixo.

(a) Pagamentos anuais

```
INPUT "Taxa de juro anual em
 %";i: DIM A(30)
FOR n=1 TO 30: LET A(n)=(1-(1
 00/(100+i))↑n)/i*100: NEXT n
FOR n=1 TO 30:PRINT TAB 5;n;T
 AB 10;A(n): NEXT n
```

(b) Pagamentos mensais

```
INPUT "Taxa de juro anual em
 %";i: LET i=i/12
DIM A(30): FOR n=1 TO 30: LET
 A(n)=(1-(100/(100+i))↑(n*12)
 )/i*100: NEXT n
FOR n=1 TO 30:PRINT TAB 5;n;T
 AB 10;A(n): NEXT n
```

A anuidade é a soma inicial que, a x% ao ano, dá um rendimento de 1 milhar de escudos por ano pelo período de n anos.

Assim, por exemplo, se tivermos 25 000 milhares de escudos para investir a 8% ao ano e quisermos levantar 15 prestações anuais idênticas, a anuidade para 15 anos a 8% será de 8,5595 milhares de escudos e 25000/8,5595=2920,74 milhares de escudos será o valor de cada prestação anual. Após os 15 anos o capital fica esgotado, pois estivemos a levantar capital e juros.

O exemplo inverso é o pagamento de empréstimos, como uma hipoteca, por exemplo, de modo a que as prestações mensais sejam constantes, isto é, inicialmente pagam-se principalmente juros, mas, gradualmente, vai-se repondo cada vez mais capital.

Para contrair um empréstimo de 20 000 milhares de escudos, amortizáveis em 25 anos a, por exemplo, 9%, dá uma anuidade calculada numa base mensal de 119,16162 milhares de escudos, ficando cada prestação mensal a 20 000/119,16 162=167,84 milhares de escudos.

PRESTACOES ANUAIS

8%

ANOS	ANUIDADES
1	0.92592593
2	1.7832647
3	2.577097
4	3.3121268
5	3.99271
6	4.6228797
7	5.2063701
8	5.7466389
9	6.2468879
10	6.7100814
11	7.1389643
12	7.536078
13	7.9037759
14	8.244237
15	8.5594787
16	8.8513692
17	9.1216381
18	9.3715871
19	9.6035992
20	9.8181474
21	10.016803
22	10.200744
23	10.371059
24	10.528758
25	10.674776
26	10.809978
27	10.935165
28	11.051078
29	11.158406
30	11.257783

PRESTACOES MENSAIS

9%

ANOS	ANUIDADES
1	11.434913
2	21.889147
3	31.446806
4	40.184783
5	48.173374
6	55.47685
7	62.153965
8	68.25844
9	73.839383
10	78.941694
11	83.605421
12	87.871093
13	91.770019
14	95.334565
15	98.59341
16	101.57277
17	104.29651
18	106.78886
19	109.06353
20	111.14495
21	113.04737
22	114.78759
23	116.37811
24	117.83222
25	119.16162

```
000006    120.37701
000007    121.48317
000008    122.59404
000009    123.44276
000010    124.28167
```


3 MELHOR RECTA DE AJUSTAMENTO

Esta sub-rotina calcula a melhor recta de ajustamento para um conjunto de pontos e traça a linha e os pontos num referencial cartesiano com eixos devidamente referenciados e com escalas apropriadas. Além disso, a rotina imprime os valores do declive (m) e da ordenada na origem (C) da recta de ajustamento e o coeficiente de correlação dos dados (r). Por fim, desenha duas linhas tracejadas para representar os limites de confiança de 95 %, os quais são também apresentados numericamente.

As primeiras doze linhas calculam os valores necessários para os cálculos posteriores a partir dos dados introduzidos. Esses valores são, nomeadamente, os valores máximos de x e de y, destinados a calcular uma escala adequada para o gráfico, e os valores médios xm e ym a partir dos quais os valores do declive e da ordenada na origem da melhor recta de ajustamento e, ainda, o valor da variância vão ser calculados utilizando as equações seguintes:

A melhor recta de ajustamento é dada por

$$y = \frac{\Sigma(x_i-\overline{x})\,(y_i\overline{y})}{\Sigma(x_i-\overline{x})^2}\cdot x + \frac{\Sigma y_i}{n} - \frac{\Sigma x_i\cdot\Sigma(x_i-\overline{x})\,(y_i-\overline{y})}{n\cdot\Sigma(x_i-\overline{x})^2}$$

onde os somatórios (Σ) se referem à variação dos valores de i de 1 a n, ou seja, $\sum_{i=1}^{n}$ representa a soma dos n valores que a expressão (escrita a seguir) toma ao variar i de 1 até n, sendo $\overline{x}$ e $\overline{y}$ os valores médios Xm e Ym.

O coeficiente de correlação é dado por

$$r = \frac{\Sigma(x_i-\overline{x})\,(y_i-\overline{y})}{\sqrt{\Sigma(x_i-\overline{x})^2\cdot\Sigma(y_i-\overline{y})^2}}$$

sendo esta equação geralmente denominada por «fórmula de

co-variância» ou «fórmula do produto momento».

As seis linhas a partir de B determinam o valor t da distribuição de Student que permite calcular os limites de confiança de 95% (±e), os quais dependem tanto do número de dados, n (também chamado «tamanho da amostra»), como do valor do desvio-padrão.

As variáveis scx e scy são factores de escala (SCale) para os gráficos, destinados a assegurar que todos os pontos possam ser situados e traçados no gráfico e que apareçam números inteiros nas escalas desenhadas sobre os eixos. Assim, se o valor máximo (de x ou de y) for inferior ou igual a 1, a escala é ampliada 10 vezes (dividindo os factores respectivos pelo mesmo valor).

A função CIRCLE é utilizada para aumentar o tamanho dos pontos na rotina de localização e traçagem. A melhor recta de ajustamento é calculada e traçada por meio da sub-rotina C e das declarações PLOT e DRAW na linha imediatamente a seguir à da instrução GO SUB C (décima segunda linha depois de B). As dez linhas que se seguem à das declarações PLOT e DRAW destinam-se a desenhar os eixos coordenados e as respectivas escalas, utilizando o ponto de coordenadas de *écran* (36,4) como origem. A décima primeira linha depois da referida imprime os vários valores (declive, ordenada na origem, etc.), servindo a declaração POKE do início da linha para imprimir o sinal ± para os limites de confiança de 95%.

As restantes linhas até C tratam das linhas tracejadas que referenciam os limites de 95%.

Como existem nove possibilidades de traçar a recta y=mx+c, dependentes do facto de o início e o fim da linha se situarem acima, dentro ou abaixo da área do *écran*, é necessário utilizar a sub-rotina C para determinar quais os valores de x1, y1, e de x2, y2, pontos entre os quais a linha deve ser traçada.

A sub-rotina D é a rotina de impressão de uma linha tracejada apresentada noutra parte do livro.


```
A  INPUT "Introduza o numero de
    pares de dados (minimo 3):
   ";n: DIM E(2,n)
   IF n<3 THEN PRINT "Dados insu
    ficientes": GO TO A
   INPUT "Introduza os nomes das
     variaveis";"Nome da abcissa
     : ";A$;"Nome da ordenada: "
   ;B$
   FOR p=1 TO n: INPUT "X";(p);"
   =";E(1,p),"y";(p);"=";E(2,p)
   : NEXT p
   LET xmx=0: LET ymx=0: LET Xm=
    0: LET Ym=0
   FOR p=1 TO n
   IF E(1,p)>xmx THEN LET xmx=E(
    1,p)
   IF E(2,p)>ymx THEN LET ymx=E(
    2,p): LET Xm=Xm+E(1,p): LET
    Ym=Ym+E(2,p): NEXT p
   LET Xm=Xm/n: LET Ym=Ym/n
   LET Sxx=0: LET Sxy=0: LET Syy
   =0: FOR p=1 TO n
   LET Sxx=Sxx+(E(1,p)-Xm)*(E(1,
   p)-Xm)
   LET Sxy=Sxy+(E(1,p)-Xm)*(E(2,
   p)-Ym)
   LET Syy=Syy+(E(2,p)-Ym)*(E(2,
   p)-Ym): NEXT p
   LET m=Sxy/Sxx: LET C=Ym-m*Xm
   LET r=Sxy/(SQR Sxx*SQR Syy)
B  DIM T(10): RESTORE B: DATA 12
   .706,4.103,3.182,2.776,2.571
   ,2.447,2.365,2.308,2.262,2.2
   28
   FOR p=1 TO 10: READ T(p): NEX
   T p
   IF n<=12 THEN LET t=T(n-2)
   IF n>=13 AND n<=27 THEN LET t
   =-.000014*(n-2)↑3+.00152*(n-
   2)↑2-.05075*(n-2)+2.5975
   IF n>=28 AND n<=62 THEN LET t
   =2.056-(n-2)*.00165
   IF n>=63 THEN LET t=1.98
   LET scx=INT (xmx/25)+1: LET s
   cy=INT (ymx/20)+1
   IF xmx<=1 THEN LET scx=scx/10
   IF ymx<=1 THEN LET scy=scy/10
   LET e=t*SQR (1-r*r)*SQR (Syy/
   (n-2))
   FOR p=1 TO n: CIRCLE 36+E(1,p
   )*8/scx,4+E(2,p)*8/scy,1: NE
   XT p
```

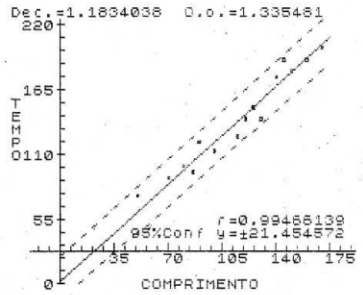
```
   LET k1=c*8/scy+4: LET k2=200*
    m*scx/scy+k1
   GO SUB C
   PLOT x1,y1: DRAW x2-x1,y2-y1
   FOR p=9 TO 29 STEP 5: PRINT A
    T 19,p;(p-4)*scx: NEXT p
   FOR p=1 TO 21 STEP 5: LET b=1
    : IF (21-p)*scy<=99 THEN LET
     b=2: IF (21-p)*scy<=9 THEN
     LET b=3
   PRINT AT p,b;(21-p)*scy: NEXT
     p
   PRINT AT 21,(35-LEN A$)/2;A$
   FOR p=1 TO LEN B$: PRINT AT p
    +(16-LEN B$)/2,0;B$(p): NEXT
     p
   PLOT 255,24: DRAW -241,0: PLO
    T 36,4: DRAW 0,163
   FOR p=20 TO 252 STEP 8: PLOT
    p,24: DRAW 0,3: NEXT p
   FOR p=76 TO 236 STEP 40: PLOT
     p,24: DRAW 0,5: NEXT p
   FOR p=12 TO 164 STEP 8: PLOT
    36,p: DRAW 3,0: NEXT p
   FOR p=4 TO 164 STEP 40: PLOT
    34,p: DRAW 5,0: NEXT p
   POKE 22325,62: PRINT AT 0,0;"
    Dec.=";m;"0,0,=";C;AT 16,19;
    "r=";r;AT 17,11;"95%Conf y="
    ;AT 17,21; OVER 1;"+";e
   LET k1=(C-e)*8/scy+4: LET k2=
    200*m*scx/scy+k1
   IF k1<0 AND k2<0 THEN GO TO P
   GO SUB C: GO SUB D
P  LET k1=(C+e)*8/scy+4: LET k2=
    200*m*scx/scy+k1
   IF k1>175 AND k2>175 THEN GO
    TO R
   GO SUB C: GO SUB D: GO TO R
C  IF k1<0 THEN GO TO E
   IF k1>175 THEN GO TO F
   LET x1=36: LET y1=k1
   IF k2<0 OR k2>175 THEN GO TO
    G
   LET x2=236: LET y2=k2: GO TO
    H
E  IF k2<0 THEN GO TO H
   IF k2>175 THEN GO TO I
J  LET x1=(85.5-85.5*SGN m-k1)/m
    *scy/scx+36: LET y1=89.5-85.
    5*SGN m: LET x2=236: LET y2=
    k2: GO TO H
F  IF k2>175 THEN GO TO H
```



```
   IF k2<0 THEN GO TO I
   GO TO J
G  LET x2=(85.5+85.5*SGN m-k1)/m
    *scy/scx+36: LET y2=89.5+85.
    5*SGN m: GO TO H
I  LET x1=(85.5-85.5*SGN m-k1)/m
    *scy/scx+36: LET y1=89.5-85.
    5*SGN m: LET x2=(85.5+85.5*S
    GN m-k1)/m*scy/scx+36: LET y
    2=89.5+85.5*SGN m
H  RETURN
D  IF x1=x2 THEN GO TO K
   IF y1=y2 THEN GO TO L
   LET p=(y2-y1)/(x2-x1)
   IF ABS p<1 THEN GO TO M
   LET p=1/p: GO TO N
L  LET p=0
M  LET i=10*SGN (x2-x1)/50R (1+p
    *p)
   FOR q=1 TO (x2-x1)/i
   PLOT x1+(q-1)*i,y1+(q-1)*i*p:
    DRAW i*.4,i*.4*p: NEXT q: RE
    TURN
K  LET p=0
N  LET i=10*SGN (y2-y1)/50R (1+p
    *p)
   FOR q=1 TO (y2-y1)/i
   PLOT x1+(q-1)*i*p,y1+(q-1)*i:
    DRAW i*.4*p,i*.4
   NEXT q: RETURN


R  RETURN
```

Exemplo
Os dados listados abaixo dão o gráfico apresentado.

COMPRIMENTO	TEMPO
50	75
70	90
80	100
100	113
115	125
140	175
170	200
150	180
130	140
86	95
90	120
160	190
145	190
120	140
125	150



4 COEFICIENTES BINOMIAIS

Esta sub-rotina calcula os coeficientes do binómio de Newton $(a+b)^n$ para qualquer n inferior a 125.

Repare-se na utilização de declarações DIM alternativas empregando um valor de t dependente da existência de uma série finita ou de uma série infinita. No caso de n não ser um número inteiro, a série é infinita e a rotina pede ao utilizador o número de termos a serem calculados.

Os coeficientes binomiais são úteis para calcular a probabilidade de ocorrer um determinado acontecimento. Por exemplo, a escolha aleatória de 3 bolas de um saco que contenha 3 bolas vermelhas e 7 bolas negras é regida pelo binómio

$$\left(\frac{3}{10}\times\frac{7}{10}\right)^3 \qquad \text{ou seja} \qquad (.3+.7)^3$$

Desenvolvendo, vem:

$$(\emptyset.3+\emptyset.7)^3=\mathbf{1}\times\emptyset.3^3+\mathbf{3}\times\emptyset.7\times\emptyset.3^2+\mathbf{3}\times\emptyset.7^2\times\emptyset.3+\mathbf{1}\times\emptyset.7^3$$

O primeiro termo do desenvolvimento dá a probabilidade de serem obtidas 3 bolas vermelhas, o segundo 2 vermelhas e 1 negra, o terceiro 1 vermelha e 2 negras e, por fim, o último termo é a probabilidade de todas as 3 bolas serem negras.

```
      INPUT n: IF ABS INT n<>n THEN
        PRINT "n nao e' um inteiro
       positivo.  Quantos termos d
       eseja? ": INPUT t: GO TO A

      DIM B(n+1): LET t=n+1: GO TO
       B
A     DIM B(t)
B     LET z=1: LET m=n
      FOR p=1 TO t: LET B(p)=z
      LET Z=Z*m/p: LET m=m-1: NEXT
       p
R     RETURN

R     FOR p=1 TO t: PRINT B(p): NEX
       T p
```

No caso de n ser um inteiro positivo e de, portanto, a série ser finita, os coeficientes binomiais (impressos a tipo negro no exemplo dado) calculados pelo programa para cada valor de n constituem uma linha do conhecido triângulo de Tartaglia-Pascal, linha cuja ordem (a contar do topo) é dada pelo valor de n. Assim, a primeira linha (um só coeficiente: 1) é dada para n=0, a segunda por n=1, etc.

5 ORDENAÇÃO DE «BOLHAS»

Esta rotina de ordenação utiliza o bem conhecido método de ordenação por comparações sucessivas, dita «ordenação de bolhas» *(bubblesort)*. Esta designação curiosa deve-se ao facto de, no mencionado método, os números elevados subirem gradualmente de posição na lista até atingirem as posições cimeiras, como bolhas de gás num líquido. A variável q destina-se a contar as trocas de posição efectuadas. A linha C decide, após comparação, se uma troca de posição entre dois números adjacentes é ou não necessária e a sub-rotina A efectua a troca, em caso afirmativo. A rotina continua a executar o ciclo até não serem necessárias mais trocas de posição, ou seja, quando q=0.

Nota. Esta rotina coloca os números de uma lista em ordem decrescente. Para que os números sejam colocados por ordem crescente, substituir o sinal > pelo sinal < na linha C.

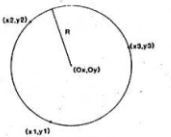
```
      INPUT n: DIM A(n): FOR p=1 TO
        n: INPUT A(p): NEXT p

B     LET q=0: FOR p=1 TO n-1
C     IF A(p+1)>A(p) THEN GO SUB A:
        LET q=q+1
      NEXT p: IF q<>0 THEN GO TO B
      GO TO R
A     LET z=A(p): LET A(p)=A(p+1):
        LET A(p+1)=z: RETURN
R     RETURN

R     FOR p=1 TO n: PRINT A(p): NEX
        T p
```

6 CIRCUNFERÊNCIA

Esta sub-rotina calcula as coordenadas do centro e o raio da única circunferência que passa por três pontos dados.

```
      INPUT "Introduza as coordenad
       as dos   tres pontos:"''"x1
       =";x1,"y1=";y1''"x2=";x2,"y2=
       ";y2''"x3=";x3,"y3=";y3


      LET D=x1*(y2-y3)+x2*(y3-y1)+x
       3*(y1-y2)
      LET OX=((x1*x1+y1*y1)*(y2-y3)
       +(x2*x2+y2*y2)*(y3-y1)+(x3*x
       3+y3*y3)*(y1-y2))/2/D
      LET OY=((x1*x1+y1*y1)*(x3-x2)
       +(x2*x2+y2*y2)*(x1-x3)+(x3*x
       3+y3*y3)*(x2-x1))/2/D
      LET R=SQR ((OX-x1)*(OX-x1)+(O
       Y-y1)*(OY-y1))
R     RETURN
R     PRINT "Raio = ";R''"Coordenad
       as do centro:"''"Ox = ";OX''
       "Oy = ";OY
```

Repare-se na particularidade de os quadrados dos valores das coordenadas serem dados por x1*x1 e não por x1↑2 devido ao facto de x1, x2, etc., poderem assumir valores negativos, o que daria origem a uma mensagem de erro se tal acontecesse utilizando ↑. Por outro lado, como uma raiz quadrada tem sempre dois valores, um positivo e um negativo, o valor de R também pode ser positivo ou negativo, devendo pois ter-se cuidado se R for utilizado na fórmula como um valor absoluto.



Circunferência

7 ARRANJOS DE +1 e −1 EM GRUPOS DE TRÊS

Esta sub-rotina gera os oito arranjos com repetição (também chamados «arranjos completos») de +1 e −1 em grupos de três e deixa o *input* inalterado.

Cada escolha pode ser feita de duas maneiras, de modo que o número total de arranjos diferentes é igual a 2*2*2=8

O primeiro arranjo é

1 1 1

Para se obter a sequência de arranjos apresentada a seguir, é trocada a posição relativa de dois números ou um número é multiplicado por −1. Assim, temos

	Arranjos			Operação efectuada
a=1	1	1	1	A(3)*(−1)
2	1	1	−1	A(2) ⟷ A(3)
3	1	−1	1	A(3)*(−1)
4	1	−1	−1	A(1)*(−1)
5	−1	−1	−1	A(3)*(−1)
6	−1	−1	1	A(2) ⟷ A(3)
7	−1	1	−1	A(3)*(−1)
8	−1	1	1	A(1)*(−1)
	1	1	1	

```
      DIM A(3): DIM P(8,3)
      LET A(1)=1: LET A(2)=1: LET A
        (3)=1
      FOR a=1 TO 8: FOR b=1 TO 3
      LET P(a,b)=A(b): NEXT b
      IF a=2 OR a=6 THEN LET C=A(3)
        : LET A(3)=A(2): LET A(2)=C:
        GO TO R
      IF a=4 OR a=8 THEN LET A(1)=-
        A(1): GO TO R
      LET A(3)=-A(3)
R     NEXT a
      RETURN
```

```
B  FOR a=1 TO 6: FOR b=1 TO 3
   PRINT P(a,b);",";: NEXT b
   PRINT : NEXT a
```


8 DECIMAL PARA BINÁRIO
(Para números inteiros positivos menores do que 256)

```
      INPUT "Introduza o numero dec
       imal: ";d

      DIM A$(8): FOR n=1 TO 8
      LET U=INT (d/2↑n-INT (d/2↑n)+
       .50005)
      LET A$(9-n)=STR$ U: NEXT n
R     RETURN

R     PRINT d,A$
```

Esta rotina é baseada no método dito das «divisões sucessivas», que consiste na divisão sucessiva por 2 (base binária) e obtenção dos respectivos restos.

Contudo, existem alguns erros nos cálculos do computador e, por isso, esta rotina não é adequada à conversão de inteiros superiores a 255. A rotina seguinte funciona melhor para esses números:

```
      INPUT "Introduza o numero dec
       imal: ";d

      LET p=1
A     IF 2↑p<ABS d+1 THEN LET p=p+1
       : GO TO A
      DIM A$(p): FOR q=1 TO p
      LET U=INT (d/2↑q-INT (d/2↑q+.
       0000001)+.5)
      LET A$(p+1-q)=STR$ U
      NEXT q
R     RETURN

R     PRINT d,A$
```

9 BINÁRIO PARA DECIMAL
(Para quaisquer números binários inteiros positivos)

Esta sub-rotina determina o número decimal equivalente a um número binário dado somando os diferentes valores para cada posição, os quais são dados pelo produto do valor absoluto (Ø ou 1) pelo valor de posição (2↑posição do dígito), sendo a primeira posição (e a última a ser calculada) zero. A soma total (d) é o número decimal pretendido.

Para números binários inferiores a 1ØØØØØØØØØØØØØØØ (65536 em decimal) pode utilizar-se, em vez da sub-rotina, o comando directo PRINT BIN b, sendo b o número em binário. Este comando dá, de imediato, o equivalente decimal de b.

```
INPUT "Introduza o numero em
 binario: ", LINE A$
```

(Utilize-se, desejando, a sub-rotina 54 — Teste para um número binário

```
    LET d=Ø: LET c=1: LET n=LEN A
    $
A   LET d=d+2↑(n-c)*VAL A$(c)
    IF n=c THEN GO TO R
    LET c=c+1: GO TO A
R   RETURN

R   PRINT A$,d
```

10 HEXADECIMAL PARA DECIMAL (Para quaisquer inteiros positivos)

A função definida pelo utilizador h(n) «desmonta» o número hexadecimal recorrendo à ordem das letras e números na lista de código ASCII do conjunto de caracteres do *Spectrum* (apêndice A do manual de programação BASIC do *Spectrum*) e efectuando a sua selecção por meio de CODE h$(n)−47. Assim, por exemplo, F tem o código (CODE) 7Ø, de modo que é o 23.º carácter (70-47) da cadeia (da 2.ª linha da rotina) a ser seleccionado, ou seja, o carácter «?», o qual tem o código 63. Subtraindo 48 a este valor, obtém-se 15, que é o equivalente decimal de F.

Ter em atenção que os símbolos <, = e > que aparecem na cadeia são os caracteres individuais «<», «=» e «>».

```
    INPUT "Introduza o numero em
     hex : ", LINE h$

    DEF FN h(n)=CODE "0123456789
     ABCDEF:;<=>?@ABCDEFGHIJKLMNO
     PQRSTUVWXYZ:;<=>?"(CODE h$(n)
     -47)-48
    LET d=0: FOR n=1 TO LEN h$
    LET d=d+FN h(n)*16↑(LEN h$-n)
     : NEXT n
R   RETURN

R   PRINT h$,d
```

11 BINÁRIO PARA HEXADECIMAL
(Para quaisquer inteiros positivos)

Existe uma relação bastante simples entre o sistema binário e o sistema hexadecimal, pois cada grupo de quatro dígitos num número binário representa um dígito do número hexadecimal equivalente. Assim, nesta sub-rotina, FN B$(s) é a lista de 16 dígitos hexadecimais, os quais são seleccionados por meio de s, sendo os números compostos por meio de grupos de 4 ciclos de q e a composição terminada quando t=n. A variável n é o comprimento, em número de dígitos, do número binário.

```
      INPUT "Introduza o numero em
       binario: ", LINE A$

      DEF FN B$(s)="0123456789ABCDE
       F"(s)
      LET n=LEN A$: DIM H$(1+INT ((
       n-1)/4)): LET t=0
A     LET S=0: LET q=0
B     LET S=S+VAL A$(n-q-INT (t/4)*
       4)*2↑q
      LET q=q+1: LET t=t+1
      IF t=n THEN LET H$(1)=FN B$(S
       +1): GO TO R
      IF q=4 THEN LET H$(LEN H$-t/4
       +1)=FN B$(S+1): GO TO A
      GO TO B
R     RETURN

R     PRINT A$,H$
```

12 DECIMAL PARA HEXADECIMAL
(Para quaisquer números inteiros positivos)

Esta rotina baseia-se também no método das «divisões sucessivas», utilizando a função definida pelo utilizador FN D$(n,d), que contém a lista de dígitos hexadecimais, os quais são seleccionados por meio dos restos das sucessivas divisões.

```
      INPUT "Introduza o numero em
       decimal: ",n


      DEF FN D$(n,d)="0123456789ABC
       DEF"(INT (n/16↑d)-16*INT (n/
       16↑(d+1))+1)
      LET A$="": FOR d=LEN STR$ n-1
       TO 0 STEP -1
      LET A$=A$+FN D$(n,d): NEXT d
      IF A$(1)="0" THEN LET A$=A$(2
       TO ): GO TO R
R     RETURN


R     PRINT n,A$
```

13 HEXADECIMAL PARA BINÁRIO
(Para quaisquer números hexadecimais inteiros positivos)

A mesma declaração (e correspondente função) DEF FN utilizada na sub-rotina de conversão hexadecimal para decimal é aqui outra vez usada para «desmontar» o número em hexadecimal. Uma série de divisões sucessivas é utilizada para determinar os sucessivos valores (Ø ou 1) de u que vão, em ordem inversa, ser alinhados em B$ para assim formar o número binário equivalente.

```
      INPUT "Introduza o numero em
       hex.: ", LINE h$

      DIM B$(4*LEN h$)
      DEF FN h(n)=CODE "0123456789
       000000:;<=>?@0000000000000000
       0000000000:;<=>?"(CODE h$(n)
       -47)-48
      FOR n=LEN h$ TO 1 STEP -1: FO
       R m=1 TO 4
      LET u=INT (FN h(n)/2↑m-INT (F
       N h(n)/2↑m)+.50005)
      LET B$(4*n-m+1)=STR$ u: NEXT
       m: NEXT n
R     RETURN

R     FOR t=1 TO 3: IF B$(t)="Ø" TH
       EN NEXT t
      PRINT h$,B$(t TO )
```

14-17 VALIDAÇÃO DE «INPUT» DE DADOS

O objectivo de uma rotina de *input* de dados é duplo. Em primeiro lugar, deve permitir que o *input* seja verificado (validado) automaticamente, de forma a assegurar que os dados que o compõem são do tipo correcto e se situam todos dentro dos limites aceites pelo programa que os vai utilizar. Esses dados podem ser, por exemplo, um número, um nome, letras isoladas, uma combinação finita de números e caracteres não numéricos, etc. Em segundo lugar, essa rotina (ou sub-rotina) deve também permitir que esses dados sejam verificados pelo operador que os introduza de modo a que este confirme se todos os valores estão correctos, se não houve engano na transcrição de palavras, etc., antes que eles sejam utilizados pelo programa. A utilização de dados não validados leva invariavelmnente a que o programa «estoire» *(crash)* em qualquer altura. Por exemplo, INPUT n$: LET n = VAL n$ provocará o bloqueio do programa e a emissão da mensagem de erro 'Nonsense in BASIC' se, acidentalmente, pressionarmos a tecla ENTER sem termos previamente digitado um carácter de cadeia. Isto pode ser evitado se, entre as duas declarações anteriores (separadas por ":"), inserimos a seguinte:

IF CODE n$=∅ THEN GO TO (n.º da linha de INPUT)

Outros testes inclusivos também podem filtrar este erro. Por exemplo:

IF n$(p)>= "∅" AND n$(p)<="9" THEN correcto

Se estivermos a definir uma variável para uma declaração DIM devemos assegurar-nos de que ela não possa tomar o valor zero. Isto pode, por exemplo, ser feito por

IF VAL n$=∅ THEN GO TO (n.º da linha de INPUT)

Uma forma generalizada de linha de INPUT é a seguinte:

N.º de linha LET q=(n.º de linha) : INPUT q$

Neste caso, q$ pode ser seguidamente testado carácter a carácter por meio de um ciclo que, em caso de erro, utilizará a instrução THEN GO TO q. No caso de não haver erro, q$ é, então, ou convertido num número por via de uma função VAL e armazenado, ou armazenado directamente, deixando as variáveis q e q$ disponíveis para o INPUT seguinte.

14 «INPUT» DE DADOS (Equações lineares com um máximo de oito variáveis)

Esta sub-rotina aceita coeficentes e constantes (termos independentes) para sistemas de equações lineares (máximo de 8), por exemplo, antes de resolver o sistema resultante.

A limitação de oito equações a oito incógnitas é posta apenas devido às dimensões do *écran*. Repare-se na utilização de AND, na terceira linha da listagem, para fornecer a versão singular e a versão plural do cabeçalho impresso antes das equações. O POKE da linha seguinte destina-se a sublinhá-lo.

As linhas 3 a 8 da sub-rotina (4 a 9 da listagem completa) imprimem as equações em forma algébrica, encadeando CHR$ (96+k) com STR$ j, "*" e CHR$ (90+k−n) de modo a formar a1*x, b2*y, etc. As linhas que se seguem a B permitem que sejam introduzidos os valores numéricos de a1, b2, etc., sendo a1*x, b2*y, etc., apagados por meio da impressão de quatro espaços em branco antes de os correspondentes coeficientes numéricos serem impressos em sua substituição. Isso é possibilitado por quatro instruções CHR$ 8, cada uma das quais faz a posição de impressão recuar de um carácter de modo que, assim, os vários valores numéricos dos coeficientes [variável e (j,k) da listagem] podem ser impressos na posição anteriormente ocupada por a1*x, etc.

Se não ficarmos satisfeitos com os valores numéricos introduzidos, podemos voltar à linha B por meio de k$ e voltar a introduzi-los, desta vez com as correcções necessárias, sendo novamente efectuado o processo de apagamento e impressão descrito anteriormente para possibilitar a substituição dos valores numéricos.

```
A   INPUT "Qual o numero de varia
      veis? ";n
    IF n<1 OR n>8 THEN GO TO A
```

```
      PRINT AT 0,6;"A equacao e':"
       AND (n=1);"As equacoes são :
        " AND (n>1)
      FOR p=6 TO 21+(n<>1)*2: POKE
       16416+p,255: NEXT p: DIM e(n
       ,n+1)
      FOR j=1 TO n: FOR k=1 TO n
      PRINT AT 2*j+2+(k>4),7*k-7-28
       *(k>4);CHR$ (96+k)+STR$ j+"*
       "+CHR$ (90+k-n)
      NEXT k: FOR k=1 TO n-1
      PRINT AT 2*j+2+(k>4),7*k-2-28
       *(k>4);"+": NEXT k
      PRINT AT 2*j+2+(k>4),25;" = "
       +"K"+STR$ j: NEXT j
      FOR k=1 TO n: PRINT AT 2,27;"
       Const";AT 2,7*k-7-28*(k>4);C
       HR$ (32+15*(k>4))+CHR$ (90+k
       -n): NEXT k
B     FOR j=1 TO n: FOR k=1 TO n+1
      INPUT "Introduza os valores n
       umericos ",(CHR$ (96+k-(k=n+
       1)*(k+21))+STR$ j);"=? ";e(j
       ,k)
      PRINT AT 2*j+2+(n>4)*(k>4),(7
       *k-7-28*(k>4))*(k<>n+1)+(k=n
       +1)*28;"      ";CHR$ 8;CHR$ 8;
       CHR$ 8;CHR$ 8;e(j,k)
      NEXT k: NEXT j
      INPUT "              OK? (s/n) ";k
       $
      IF k$<>"s" THEN GO TO B
R     RETURN
```

Exemplos

```
n = 5
```

As equacoes sao :

V/Z	W	X	Y	Const
a1*V + b1*W + c1*X + d1*Y + e1*Z				= K1
a2*V + b2*W + c2*X + d2*Y + e2*Z				= K2
a3*V + b3*W + c3*X + d3*Y + e3*Z				= K3
a4*V + b4*W + c4*X + d4*Y + e4*Z				= K4
a5*V + b5*W + c5*X + d5*Y + e5*Z				= K5


```
Introduza os valores numericos
a1=?
```

As equacoes sao :

V/Z	W	X	Y	Const
15	+ 25	+ 9.5	+ -5	+
7				= 27
9.6	+ -10	+ 4.3	+ 7	+
2				= 51
3.7	+ 16	+ 6.9	+ 13	+
4				= 97
16	+ 91	+ 2	+ 81	+
6				= 115
6	+ 2	+ 15	+ 6	+
8				= 23

OK? (s/n)

15 «INPUT» DE DADOS (Matrizes ou quadros)

Esta sub-rotina permite que uma matriz ou quadro *(array)* de números seja introduzida e verificada.

A apresentação inicial é, como na sub-rotina anterior, também em formato algébrico, com os parênteses rectos adequados. Repare-se que a posição desses parênteses é determinada utilizando os valores finais de k e de j (isto é, n+1 e m+1). A partir da linha A, a sub-rotina permite que os valores numéricos sejam introduzidos, imprimindo-os em substituição das suas representações algébricas a11, a12, etc. Se tivermos cometido um erro ao introduzir os valores, a sub-rotina permite-nos voltar a introduzi--los, com as respectivas alterações, acrescentando contudo um «outra vez, correctamente» à mensagem indicativa de quando devemos introduzir os valores. Este acrescentamento é efectuado quando a variável de cadeia k$ se torna diferente de «s», carácter atribuído originalmente à variável logo depois da declaração DIM.

```
    INPUT "Introduza as dimensoes
      da matrizna forma de m linh
     as n colunas :"'"m=";m;,"n=
     ";n

    PRINT AT 1,8;"A matriz é:"
    FOR j=1 TO 12: POKE 16455+j,2
     55: NEXT j
    DIM a(m,n): LET k$="s"
    FOR j=1 TO m: FOR k=1 TO n
    PRINT AT 3+2*j,4*k-3;"a"+STR$
      j+STR$ k
    NEXT k: NEXT j
    PLOT 10,140: DRAW -8,0: DRAW
     0,16-16*j: DRAW 8,0
    PLOT 32*k-38,140: DRAW 8,0: D
     RAW 0,16-16*j: DRAW -8,0
A   FOR j=1 TO m: FOR k=1 TO n
    INPUT "Introduza os valores n
     umericos ";"outra vez, corr
     ectamente" AND (k$<>"s");';"
```

```
    a"+STR$ j+3+STR$ k;" = ";a(j,k
    )
  PRINT AT 2*j+3,4*k-3;"    ";CH
   R$ 8;CHR$ 8;CHR$ 8;a(j,k)
  NEXT k: NEXT j
  INPUT "                OK? (s/n) ";k
  $
  IF k$<>"s" THEN GO TO A
R RETURN
```

Exemplo

m=5 n=7

A matriz é:

$$\begin{bmatrix} a11 & a12 & a13 & a14 & a15 & a16 & a17 \\ a21 & a22 & a23 & a24 & a25 & a26 & a27 \\ a31 & a32 & a33 & a34 & a35 & a36 & a37 \\ a41 & a42 & a43 & a44 & a45 & a46 & a47 \\ a51 & a52 & a53 & a54 & a55 & a56 & a57 \end{bmatrix}$$

```
Introduza os valores numericos
a11 =
```

A matriz é:

$$\begin{bmatrix} 37 & 16 & 9 & 5 & 4 & 71 & 10 \\ 106 & 5 & 0 & 6 & 23 & 17 & 9 \\ 12 & -6 & 5 & 3 & 17 & 9 & 12 \\ 2 & -5 & 6 & 9 & 18 & 21 & 3 \\ 51 & 6.5 & 3.4 & 2.2 & 91 & 12 & 15 \end{bmatrix}$$

```
OK? (s/n)
```


16 «INPUT» DE DADOS (Variável simples)

Esta sub-rotina destina-se à introdução de dados para análise estatística.

Com variáveis simples ocorre frequentemente o caso de haver demasiados dados para poderem ser apresentados simultaneamente. Nesta rotina, os dados são todos armazenados inicialmente em V(n). Contudo, cada um deles é previamente testado pela sub-rotina C (SUB C), que verifica as suas características numéricas. Se o texte resultar positivo, t atingirá o valor de LEN V$+1 e o dado V$(p) testado é transformado em valor numérico pela função VAL.

Os valores são apresentados no *écran* utilizando a emissão da mensagem «scroll?» para controlar o caudal de dados. A variável p destina-se também a verificar se um dado que queiramos corrigir existe na realidade (pode dar-se o caso de o conjunto de dados original conter, por exemplo, 25 valores e, por engano, pedirmos para corrigir o 29.º ou qualquer outro que não conste do conjunto). A sub-rotina B faz o programa esperar que tomemos uma decisão.

```
0   INPUT "Qual o numero de dados
    ? ";n
    IF n)=4 THEN GO TO A
    PRINT n;" dados é um numero i
     nsuficientepara efectuar uma
     analise este- tistica corre
    cta.";//"Pressione qualquer t
    ecla para  recomecar.": GO
    SUB 220: CLS
    GO TO 0
A   DIM V(n)


    FOR p=1 TO n
E   INPUT "Introduza os dados: V"
    ;(p);"="; LINE V$: GO SUB C
    IF t=LEN V$+1 THEN LET V(p)=V
     AL V$: GO TO F
```

```
    GO SUB D: GO TO E
F   NEXT p
M   PRINT "Estes dados estao corr
     ectos? "
    GO SUB G: GO TO H
G   PRINT : FOR p=1 TO n: PRINT "
     V";p;"=";V(p): NEXT p: RETUR
     N
H   INPUT "(s/n)? ";k$
    IF k$="s" THEN GO TO R
K   INPUT "Que dado quer alterar-
     Linha? "; LINE V$: GO SUB C:
     IF t=LEN V$+1 THEN LET p=VAL
      V$: GO TO J
    GO SUB D: GO TO K
J   IF p<=n THEN GO TO L
    PRINT "Esse numero nao fazia
     parte dos dados originais. P
     rima qualquer tecla para vol
     tar atras.": GO SUB B: GO TO
      K
L   INPUT "Introduza o valor corr
     ecto para V";(p);"=? "; LINE
      V$: GO SUB C: IF t=LEN V$+1
     THEN LET V(p)=VAL V$: CLS :
     GO TO M
    GO SUB D: GO TO L
B   IF INKEY$<>"" THEN GO TO B
@   IF INKEY$="" THEN GO TO @
    RETURN
C   FOR t=1 TO LEN V$: IF V$(t)>=
     "0" AND V$(t)<="9" OR V$(t)=
     "," OR V$(t)="+" OR V$(t)="-
     " THEN NEXT t
    CLS : RETURN
D   PRINT V$;" nao e'um numero":
     RETURN
R   RETURN
```

Exemplo

```
Qual o numero de dados? 12

Introduza os dados: V1=

Estes dados estao correctos?

V1=12.1
V2=23.05
V3=34.3
V4=44
V5=5.6
V6=67
V7=78
V8=89
V9=12
V10=2.3
V11=34
V12=45


(s/n)? "n"

Que dado quer alterar-Linha? 5

Introduza o valor correcto para
V5=?
```

17 «INPUT» DE DADOS
(Coordenadas x e y, dados estatísticos, etc.)

O problema da existência de uma grande quantidade de dados é, neste caso, superado tratando-os em grupos de dez de cada vez. O espaço de memória reservado por meio de DIM B para conter os dados é calculado de forma a conter um número de pares de valores que seja múltiplo de 10, número esse que pode bem ser superior ao estritamente necessário para conter a totalidade dos dados. Os dados são armazenados e apresentados no *écran* para verificação visual, em grupos de 10, utilizando t como contador de blocos.

É feita uma verificação a qualquer número de linha do conjunto de dados na qual eventualmente queiramos efectuar uma correcção, para confirmar se o par de valores que a constituem está efectivamente no *écran*. Depois das eventuais correcções terem sido efectuadas (ou da verificação visual por nós efectuada não ter detectado erros), os dados são transferidos na sua totalidade para E(2,n), quadro este que tem já as dimensões exactas para os conter.

Assim, esta sub-rotina aceita pares de valores, permite a sua verificação e eventual correcção e, por fim, armazena-os num espaço de memória (definido por um quadro bidimensional) reservado para o efeito.

```
      INPUT "Numero de pares de dad
       os: ";n

      DIM B(2,10*(INT ((n-1)/10)+1)
       ): FOR p=1 TO n
      INPUT "Introduza os valores d
       as coorde-nadas x e y:"/"x"
       ;(p);"=";B(1,p),"y";(p);"=";
       B(2,p)
      NEXT p: LET t=0
A     FOR p=t TO INT ((n-1)/10): LE
       T t=p
      FOR q=1 TO 10
      PRINT AT q+3,4;"    ";AT q+3,
```

```
    20;"     "
   PRINT AT q+3,1;"x";q+10*p;"="
    ;B(1,q+10*p);" AT q+3,17;"y"
    ;q+10*p;"=";B(2,q+10*p)
   NEXT q
   INPUT "Dados correctos (s/n)?
    ";k$
   IF k$="s" THEN GO TO B
D  INPUT "Que linha, entre as li
    nhas ";(1+10*p);" e";(10*(p
    +1));" deseja alterar?";"Lin
    ha - ";l
   IF l>=1+10*p AND l<=10*(p+1)
    THEN GO TO C
   PRINT "Numero de linha incorr
    ecto": GO TO D
C  INPUT "Introduza os valores c
    orrectos: x";(l);"=";B(1,l),
    "y";(l);"=";B(2,l)
   GO TO A
B  NEXT p
   DIM E(2,n): FOR p=1 TO n
   LET E(1,p)=B(1,p): LET E(2,p)
    =B(2,p): NEXT p
   DIM B(1)
R  RETURN
```

Exemplo

```
x11=32.7          y11=45.9
x12=34.7          y12=12.8
x13=30            y13=12.9
x14=27            y14=19.6
x15=34.5          y15=13.6
x16=12.6          y16=12.7
x17=23.7          y17=12
x18=23            y18=67
x19=22            y19=56.8
x20=12.5          y20=34.8


Dados correctos (s/n)? "n"

Que linha, entre as linhas 11 e
20 deseja alterar?

Linha - 18
```



```
Introduza os valores correctos:
x18=23.5          y18=67.5


x11=32.7          y11=45.9
x12=34.7          y12=12.8
x13=30            y13=12.9
x14=27            y14=19.6
x15=34.5          y15=13.6
x16=12.6          y16=12.7
x17=23.7          y17=12
x18=23.5          y18=67.5
x19=22            y19=56.8
x20=12.5          y20=34.8


Dados correctos (s/n)? "s"


x21=15.9          y21=40.3
x22=21.3          y22=37.5
x23=9.8           y23=19.8
x24=4.2           y24=62
x25=3.1           y25=15.9
x26=1.6           y26=32.3
x27=0.2           y27=9.6
x28=0             y28=0
x29=0             y29=0
x30=0             y30=0


Dados correctos (s/n)? " "
```

18 FICHEIRO DE IMAGEM

A informação destinada a ser apresentada no *écran* de TV é armazenada nos *bytes* com os endereços 16384 a 22527 da RAM. Isto inclui as duas últimas linhas do *écran*, normalmente reservadas às declarações de INPUT e a mensagens emitidas pelo computador (mensagens de erro, etc.), e que não fazem parte da zona de *output* normal do *écran*. A figura relativa ao ficheiro de imagem mostra as relações existentes entre os números de linha e de coluna utilizados nas declarações PRINT AT, ou entre as coordenadas x e y das declarações PLOT, CIRCLE e DRAW, e a célula ou posição de memória do *byte* que representa a fila superior de 8 *pixels* de qualquer posição de impressão (também chamada «posição de carácter» ou «célula de carácter»).

O ficheiro de imagem é dividido em três blocos de oito linhas, cada uma delas com 32 posições de impressão (quadrados, na figura) tendo, por sua vez, cada uma dessas posições de impressão oito *bytes* a constituí-la, representando cada *byte* uma fila de 8 *pixels* (*picture elements* ou elementos de imagem). Em cada bloco citado, as posições de memória dos *bytes* são ordenadas de forma a que os sucessivos *bytes* que constituem a primeira fila de *pixels* de cada uma das 32 posições de impressão de uma linha sejam adjacentes em sequência (isto é, as posições de memória por eles ocupadas têm endereços sucessivos). Os endereços dos *bytes* da primeira fila de *pixels* são consecutivos ao longo das 8 linhas do bloco, seguindo-se, em continuação da sequência, os da segunda fila e assim por diante, até ao final da oitava fila de *pixels*. Assim, o endereço do *byte* correspondente à segunda fila de *pixels* de uma posição de impressão ou de carácter é superior em 256 (32 *bytes*×8 linhas) ao da primeira fila da mesma posição de carácter, e assim por diante, dentro do mesmo bloco. Com esta disposição, são primeiramente armazenados os 256 *bytes* da primeira fila de *pixels* das 8 linhas do primeiro bloco, depois os 256 da segunda fila e assim por diante, até todo o bloco ter sido acomodado. O processo é então repetido para o segundo bloco e, depois, para o terceiro.

Todo este processo parece bastante complicado, mas a sua visualização torna-se clara se fizermos executar o seguinte, em modo directo (sem número de linha, seguido apenas de ENTER):

FOR i=16384 TO 23295: POKE i,41: NEXT i: PAUSE Ø

Observe-se a ordem por que as coisas acontecem, após termos introduzido a linha acima. A declaração PAUSE Ø no final destina-se a que o computador não apague as duas últimas linhas do terceiro bloco para imprimir a mensagem «Ø OK, Ø:3», indicando que terminou, correctamente, a execução do que lhe foi pedido. Se, depois, pressionarmos qualquer tecla, a mensagem aparecerá, embora com a indicação Ø:4 (devido à quarta declaração — PAUSE Ø), no espaço das citadas duas últimas linhas do *écran*, correspondentes ao mencionado espaço reservado ao INPUT e a mensagens do computador.

A relação entre *bytes* adjacentes é apresentada na figura do ficheiro de imagem. Os pontos chave são os seguintes:

1) Os quadrados adjacentes (posições de impressão) diferem de 1.

2) Os quadrados adjacentes verticalmente possuem as respectivas primeiras filas de *pixels* (representadas por um *byte*, cada) com endereços que diferem de 32, excepto na transição de um bloco para outro.

3) Filas adjacentes de um mesmo quadrado têm endereços que diferem de 256, como explicámos anteriormente.

4) O último (oitavo) *byte* de um quadrado tem um endereço superior em 175Ø ao do primeiro *byte* do quadrado imediatamente inferior, excepto na transição entre um bloco e outro.

5) Na transição entre dois blocos, os primeiros *bytes* de dois quadrados adjacentes verticalmente têm endereços que diferem de 1824 e os dos *bytes* adjacentes (último e primeiro, respectivamente) desses quadrados diferem de 32.

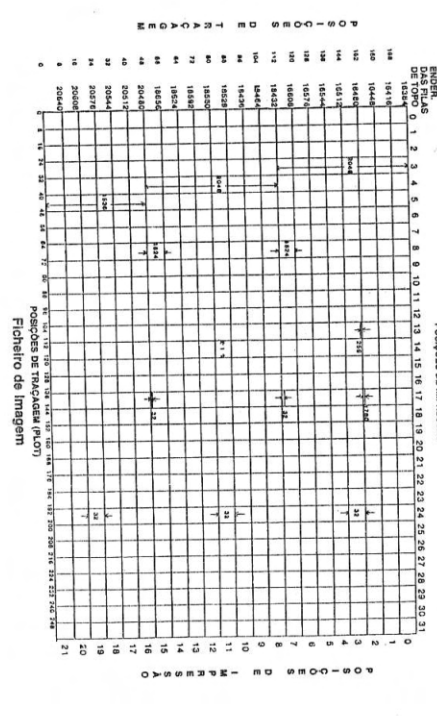
6) Existem 2Ø48 *bytes* em cada um dos dois primeiros blocos e 1536 no terceiro, que possui apenas 6 linhas, se não contarmos com as duas linhas da zona de INPUT e de mensagens, reservadas para esse fim.

A relação entre as posições de linha (y) e de coluna (x) de qualquer posição de impressão (quadrado) do *écran* e o endereço do primeiro *byte* desse quadrado é dada por:

end=16384+32*(y+56*INT(y/8))+x

onde x e y são os mesmos que seriam utilizados numa declaração PRINT AT X,Y.

Os outros *bytes* podem ser localizados de acordo com a relação dada utilizando o diagrama da figura ou os pontos-chave listados acima. É depois fácil descobrir em que endereços fazer POKE para estabelecer um sincronismo com a informação relativa a declarações PRINT AT ou PLOT, etc. (informação relativa a posições de impressão ou posições de «traçagem», respectivamente).



As variáveis de sistema com os endereços 23688 (S. POSN) e 23689 permitem determinar a posição em que, no *écran*, se encontra a posição de impressão corrente. Assim, PEEK 23688 e PEEK 23689 permitem obter informação sobre a localização, respectivamente em termos de coluna (x) e de linha (y), da posição de impressão e isto pode ser utilizado em combinação com a informação apresentada no diagrama do ficheiro de imagem para localizar correctamente os endereços a utilizar em instruções POKE. Por exemplo:

```
DEF FN U()=17216-PEEK 23688-32*(PEEK
23688=33)-32*PEEK 23689+1792*INT ((25-PEEK
23689)/8)
```

permite que qualquer carácter seja sublinhado imediatamente após a sua impressão. Podemos utilizar também uma expressão semelhante, empregando PEEK 23684+256*PEEK 23685 (variável de sistema DF CC), a qual nos dá o endereço, no ficheiro de imagem, da posição de impressão corrente.

Experimentemos a seguinte rotina de exemplo:

```
  10 LET SOMA=547250.5: LET S$=S
TR$ INT SOMA+"$"+STR$ INT ((SOMA
-INT SOMA)*10)+"0"
  20 DEF FN U()=17216-PEEK 23688
-32*(PEEK 23688=33)-32*PEEK 23689
9+1792*INT ((25-PEEK 23689)/8)
  30 PRINT : PRINT
  40 PRINT TAB 15;"Total ";: FOR
P=1 TO LEN S$: PRINT S$(P);: PO
KE FN U(),255: POKE FN U(),25
5: NEXT p
```

Deveremos obter 547250$50

Combinada com outra declaração DEF FN para alinhamento decimal dos números (cifrões uns debaixo dos outros), esta rotina pode constituir uma rotina de impressão ordenada de folhas de balanço. Exemplo:

DEF FN D$(a,n)="—— 32 espaços em branco ——" (TO a −LEN STR$ INT n)

19 IMPRESSÃO EM TAMANHO DUPLO

Esta sub-rotina imprime os caracteres ASCII do *Spectrum* num formato de 16×16 *pixels*, tamanho duplo portanto, mas tem o inconveniente de ser de execução lenta.

A primeira linha da sub-rotina propriamente dita (declarações LET) é indispensável para evitar que ocorra o caso de metade de uma letra ser impressa no extremo direito do *écran* e a outra metade do extremo esquerdo, sobre a linha seguinte. A segunda linha é igualmente importante, pois, dado haver apenas espaço para 176 posições de impressão em tamanho duplo (704/(2×2)), poderia bem acontecer estarmos a fazer POKE no ficheiro de atributos! A sub-rotina funciona lendo, por meio de PEEK, o conteúdo da célula de memória RAM com o endereço dado por 15360+8*CODE A$(n), que armazena a informação sobre o padrão de *pixels* do carácter A$(n). Cada um desses números é então transformado em quatro números diferentes, recorrendo à tabela de consulta T(2,16), e estes são nesse momento introduzidos por meio de POKE nas células de memória (correspondentes a posições de carácter) adequadas (ver Ficheiro de imagem).

```
A  INPUT "Introduza o que quer i
    mprimir:";A$
   INPUT "Introduza a posicao em
    que"'"a impressao deve come
    car:"'"Linha numero?  ";H,"C
    oluna numero? ";V

   LET H=2*INT (H/2): LET V=2*IN
    T (V/2)
   IF LEN A$>176-8*H-V/2 THEN PR
    INT "Input inaceitavel": GO
    TO A
   DIM T(2,16): FOR p=1 TO 16
   LET T(1,p)=p-1: READ T(2,p):
    NEXT p
B  RESTORE B: DATA 0,3,12,15,48,
    51,60,63,192,195,204,207,240
    ,243,252,255
   LET I=16384+32*H+V
   FOR n=1 TO LEN A$: IF A$(n)="
```

```
" THEN GO TO D
LET Ende=15360+3*CODE A$(n)
FOR p=0 TO 7: LET t=PEEK (End
e+p): IF t=0 THEN GO TO C
FOR Z=1 TO 16: IF T(1,Z)=t-16
*INT (t/16) THEN LET t1=T(2,
Z)
IF T(1,Z)=INT (t/16) THEN LET
t2=T(2,Z)
NEXT Z
LET s=2*(n-1)+512*p-INT (p/4)
*2016+32*INT ((U+2*n-1)/32)+
1792*INT ((8*H+V/2+n-1)/64)
POKE I+s,t2: POKE I+s+1,t1: P
OKE I+s+256,t2: POKE I+s+257
,t1
C NEXT P
D NEXT n
R RETURN
```

```
*IMPRESSAO EM
TAMANHO DUPLO
```

```
*Impressao em
tamanho duplo
```


20 DESENHAR LINHAS ENTRE DOIS PONTOS

Apesar de ser fácil desenhar no *écran* uma linha contínua, recorrendo aos comandos PLOT e DRAW, surgem dois problemas quando se trata de uma linha tracejada. Em primeiro lugar, ao tratar quer de linhas verticais, quer de linhas horizontais, qualquer rotina simples (que trate ambos os casos da mesma forma) produzirá com alguma delas a emissão da mensagem de erro «Number too big» (número grande de mais), acompanhada de correspondente «estoiro» *(crash)* ou bloqueio. Assim, são necessárias duas rotinas distintas, uma para cada caso. O segundo problema consiste em manter a mesma relação de comprimento traço-espaço em linhas desenhadas em ângulos diferentes. A linha A, juntamente com as quatro seguintes, trata do caso de uma linha horizontal, enquanto a linha B (e as quatro seguintes) se ocupa de uma linha vertical. A variável p representa o declive mas, caso o valor dessa variável seja inferior a 1, é utilizado o valor do inverso de p (1/p) em vez do valor de p. A variável i representa o intervalo entre cada novo início de traçagem (entre o início de um segmento de tracejado e o início do seguinte), e é estabelecido um ciclo sob controle da variável n para localizar o ponto x1,y1 (PLOT x1,y1) e depois desenhar um traço por 1/2.5 (DRAW .4) da distância i, de x1,y1 ao ponto de traçagem (PLOT) seguinte, e assim por diante. A função SGN nas declarações de definição de i (LET i= . . .) determina o sentido da traçagem (da esquerda para a direita ou vice-versa, conforme o sinal de declive) e a função SQR (1+p*p) mantém a relação traço-espaço constante para vários declives da linha tracejada.

a) Esta sub-rotina desenha uma linha a cheio entre dois pontos de coordenadas (x1,y1) e (x2,y2), sendo:
$\emptyset <= x1 <= 255$, $\emptyset <= x2 <= 255$, $\emptyset <= y1 <= 175$, $\emptyset <= y2 <= 175$

```
INPUT "x1=";x1;" y1=";y1,"x2=
 ";x2;" y2=";y2

PLOT x1,y1: DRAW x2-x1,y2-y1
```

b) Esta sub-rotina desenha uma linha tracejada entre dois pontos, de coordenadas (x1,y1) e (x2,y2), sendo:

$\emptyset <= x1,x2 <= 255 \quad \emptyset <= y1,y2 <= 175$

```
    INPUT "x1=";x1;" y1=";y1,"x2=
     ";x2;" y2=";y2

    IF x1=x2 THEN GO TO A
    IF y1=y2 THEN GO TO B
    LET p=(y2-y1)/(x2-x1)
    IF ABS p<1 THEN GO TO C
    LET p=1/p: GO TO D
B   LET p=0
C   LET i=10*SGN (x2-x1)/SQR (1+p
     *p)
    FOR n=1 TO (x2-x1)/i
    PLOT x1+(n-1)*i,y1+(n-1)*i*p:
     DRAW i*.4,i*.4*p
    NEXT n: GO TO R
A   LET p=0
D   LET i=10*SGN (y2-y1)/SQR (1+p
     *p)
    FOR n=1 TO (y2-y1)/i
    PLOT x1+(n-1)*i*p,y1+(n-1)*i:
     DRAW i*.4*p,i*.4
    NEXT n
R   RETURN
```


21 CÁLCULO DE UM DETERMINANTE

Esta elegante sub-rotina baseia-se no facto de, no método de inversão de matrizes de Gauss-Jordan (ver Inversão de matrizes), o elemento do canto inferior direitó da matriz, no final de cada ciclo, até ao último, ter o mesmo valor do determinante procurado dividido pelo menor complementar desse elemento. Para quem não está familiarizado com esta nomenclatura, podemos dizer que, dada uma matriz de ordem n, com um respectivo determinante d da mesma ordem, se formarmos uma matriz com os elementos da intersecção das primeiras k linhas e k colunas da matriz dada (sendo, é claro k<=n), obteremos uma matriz (digamos, uma «submatriz» da matriz original) de ordem k, cujo determinante se designa por «menor de ordem k» do determinante d. O «menor complementar» *do elemento citado* será o determinante da matriz de ordem n-1 formada com os elementos da intersecção das primeiras n-1 linhas com as primeiras n-1 colunas da matriz de ordem n dada (ou seja, será o determinante da matriz original desprovida da última coluna da direita e da última linha). Não sabemos o valor deste menor, mas ele pode ser calculado da mesma forma, considerando um menor com duas colunas e duas filas a menos do que a matriz original. Repetindo este processo n-1 vezes, ficaremos por fim com um menor representado apenas pelo elemento D(1,1), do canto superior esquerdo da matriz original, cujo valor conhecemos. A partir daqui, a resposta é o produto de cada um destes passos.

A primeira linha da sub-rotina propriamente dita (terceira da listagem) evita que esta «estoire» na linha 5 [caso $A(i,i)=\emptyset$]. adicionando duas linhas na sub-rotina A (isto não altera o valor do determinante). A(a,a) é um conjunto de determinantes (cada um deles mais pequeno que o anterior), o qual é invertido por meio das 10 linhas seguintes (ver Inversão de matrizes), embora o processo seja interrompido um ciclo antes do fim pela linha 5, a qual faz i variar de 1 até a-1. A décima terceira linha (anterior à linha A) calcula o produto dos resultados de cada inversão. Na

nona linha aparece uma subtracção da qual pode resultar o valor zero, valor esse que, se atribuído a um elemento de uma diagonal da matriz (A(i,i)), iria provocar no ciclo seguinte o mencionado «estoiro», da rotina na segunda declaração da linha 5, com a emissão da mensagem de erro «Number too big» (número grande de mais) resultante da tentativa da divisão de um número por zero. As linhas C e B verificam a possibilidade de tal acontecer e, caso afirmativo, fazem a sub-rotina «abortar» (interromper o curso de execução), pois o valor do determinante procurado é, nesse caso, zero. Isto sucede, nomeadamente, se duas linhas ou duas colunas forem iguais, se uma delas for múltipla de outra (ver segundo exemplo), se todos os elementos de uma delas forem zero ou ainda se qualquer delas for combinação linear das restantes.

Esta sub-rotina calcula pois o valor numérico de um determinante |A| de ordem n (de uma matriz da mesma ordem).

```
    INPUT "Ordem do determinante?
      ";n: DIM D(n,n)
    FOR i=1 TO n: FOR j=1 TO n
    INPUT "D(";i;",";j;")=";D
      (i,j): NEXT j: NEXT i

    IF D(1,1)=0 THEN GO SUB A
    LET D=D(1,1): FOR a=n TO 2 ST
      EP -1: DIM A(a,a)
    FOR i=1 TO a: FOR j=1 TO a: L
      ET A(i,j)=D(i,j)
    NEXT j: NEXT i
    FOR i=1 TO a-1: LET A(i,i)=1/
      A(i,i)
    FOR j=1 TO a: IF j=i THEN GO
      TO B
    LET A(j,i)=A(j,i)*A(i,i)
    FOR k=1 TO a: IF k=i THEN GO
      TO C
    LET A(j,k)=A(j,k)-A(j,i)*A(i,
      k)
C   NEXT k
B   NEXT j: IF ABS A(i+1,i+1)<=1E
      -8 THEN GO TO E
    NEXT i
    LET D=D*A(a,a): NEXT a: GO TO
      R

A   FOR i=2 TO n: IF D(1,i)<>0 TH
      EN GO TO D
    NEXT i
E   LET D=0: GO TO R
D   FOR j=1 TO n: LET D(j,1)=D(j,
      1)+D(j,i): NEXT j
    RETURN
R   RETURN

R   PRINT "Det D=";D
```


Exemplos

n=4

$$\begin{vmatrix} -2 & 4 & 7 & 3 \\ 8 & 2 & -9 & 5 \\ -4 & 6 & 8 & 4 \\ 2 & -9 & 3 & 8 \end{vmatrix}$$

```
Det D=2140
```

n=5

$$\begin{vmatrix} 15 & -7 & 6 & 9 & 3 \\ 2 & 4 & 6 & 8 & 10 \\ 31 & 6 & 11 & 17 & 2 \\ 1 & 2 & 3 & 4 & 5 \\ 81 & 9 & 23 & 1 & 3 \end{vmatrix}$$

```
Det D=0
```

```
  5 REM ************************
          CALCULO DE UM
          DETERMINANTE
        ************************
 10 INPUT "Ordem do determinant
e? ";n: DIM D(n,n)
 20 FOR i=1 TO n: FOR j=1 TO n
 30 INPUT "D(";i;",";j;")=";D(i,j): NEXT j: NEXT i
 35 GO SUB 500: REM ***********
 40 IF D(1,1)=0 THEN GO SUB 170
 50 LET D=D(1,1): FOR a=n TO 2
STEP -1: DIM A(a,a)
 60 FOR i=1 TO a: FOR j=1 TO a:
 LET A(i,j)=D(i,j)
 70 NEXT j: NEXT i
 80 FOR i=1 TO a-1: LET A(i,i)=
1/A(i,i)
 90 FOR j=1 TO a: IF j=i THEN G
O TO 140
 100 LET A(j,i)=A(j,i)*A(i,i)
 110 FOR k=1 TO a: IF k=i THEN G
O TO 130
 120 LET A(j,k)=A(j,k)-A(j,i)*A(
i,k)
 130 NEXT k
 140 NEXT j: IF ABS A(i+1,i+1)<=
1E-8 THEN GO TO 190
 150 NEXT i
 160 LET D=D*A(a,a): NEXT a: GO
TO 220
 170 FOR i=2 TO n: IF D(1,i)<>0
THEN GO TO 200
 180 NEXT i
 190 LET D=0: GO TO 220
 200 FOR j=1 TO n: LET D(j,1)=D(
j,1)+D(j,i): NEXT j
 210 RETURN
 220 PRINT '''"Det D=";D
 230 STOP
 500 REM ************************
          Rotina de impressao 3D
        ************************
 510 FOR i=1 TO n: FOR j=1 TO n
 520 PRINT AT 2*i,4*j-3;D(i,j)
 530 NEXT j: NEXT i
 540 PLOT 2,167: DRAW 0,8-16*i
 550 PLOT 32*j-22,167: DRAW 0,8-
16*i
 560 RETURN
```


22 FACTORIAL DE N

Esta sub-rotina calcula o factorial de n, também designado por «n factorial», para qualquer n inteiro positivo menor do que 34.

A sub-rotina apresenta um ciclo extremamente simples que efectua as multiplicações sucessivas necessárias ao cálculo do factorial.

Para quem não sabe o que é um factorial, damos os seguintes exemplos elucidativos do factorial de n (representado por n!):

n=5 5!=5×4×3×2×1=120
n=7 7!=7×6×5×4×3×2×1=5040
n=1 1!=1

```
      INPUT "Introduza n (n<34): ";
       n

      LET z=1: FOR m=n TO 1 STEP -1
       : LET z=z*m: NEXT m
R     RETURN

R     PRINT n;"!=";z
```

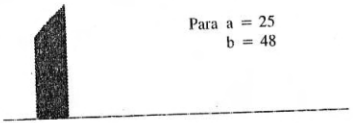
23 OPERADOR «HEAVISIDE»

Esta sub-rotina executa a denominada «operação *heaviside*», que consiste em «ligar» uma função em x=a e «desligá-la» em x=b (isto é, entre a e b, o valor de y é dado pela função, sendo zero ou qualquer outro valor constante para os restantes valores de x).

Isto é muito útil no estudo de problemas de mecânica e de electricidade e electrónica onde ocorram funções escalão, como, por exemplo, num circuito com um interruptor que é ligado ou desligado. É muitas vezes utilizada uma função escalão para testar a estabilidade de um sistema ou a sua resposta. No exemplo (segunda listagem), é dada a resposta de um circuito RC, contendo um condensador e uma resistência em série, a um impulso de tensão de forma rectangular e amplitude constante (também chamada «onda quadrada»), formado por ligações e cortes instantâneos e sucessivos de uma tensão ou voltagem aplicada ao circuito. Repare-se que, apesar de a tensão de entrada ser ligada e cortada bruscamente (sinal desenhado na parte inferior do *écran*), a tensão aos terminais do condensador leva sempre um certo atraso de resposta (sinal desenhado na parte superior do *écran*), pois este leva tempo a carregar e a descarregar. Assim, a tensão aos terminais do condensador aparece sob a forma de uma série de curvas de carga e descarga, formando uma denominada «onda quadrada amortecida» (isto, é claro, no caso presente, em que o condensador dispõe do tempo necessário para se carregar e descarregar). Vale a pena alterar o valor de RC e observar o efeito na resposta do circuito. Este teste é muitas vezes utilizado para testar, por exemplo, altifalantes de colunas de alta fidelidade.

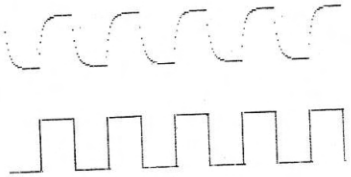
```
DEF FN A(X)=SQR X: REM POR
 EXEMPLO
INPUT a,b

DEF FN H(X,a,b)=(X)=a)-(X)=b)
DEF FN B(X)=FN A(X)*FN H(X,a,
 b)
```

```
R  RETURN
R  FOR X=0 TO 255
   PLOT X,0
   DRAW 0,10*FN B(X)
   NEXT X
```



Exemplo — Onda quadrada e onda quadrada amortecida

```
LET a=25: LET b=48: LET RC=3
DEF FN H(x,a,b)=(x>=a)-(x>=b)
FOR X=1 TO 250: LET t=(X-25*I
 NT (X/25))/RC
PLOT X,64+32*EXP -t+32*FN H(x
 ,a,b+2)*(1-2*EXP -t)
PLOT X,32*FN H(X,a+1,b+1)
IF X=a-1 OR X=b+1 THEN DRAW 0
 ,32
IF X>b THEN LET a=a+50: LET b
 =b+50
NEXT X
```



24-27 CICLOS

Os ciclos *(loops)* são um meio poderoso para gerar e manipular dados, e estes exemplos são apresentados para mostrar algumas das coisas que podem ser feitas com eles. No ciclo separador, quando a variável de controle de ciclo a excede o valor 4, o controle do ciclo passa a fazer-se pela segunda declaração FOR. No ciclo misto aparecem em acção duas operações de ciclo, comandadas, respectivamente, pelas declarações FOR a=1 TO 7 e LET a=a+1, tendo esta última a função de permitir o cálculo apenas de 8 arranjos de a e b em vez de todos os 28 possíveis com a disposição indicada. O termo (a=4) destina-se a ajustar o valor de a no ponto de transição (ao fim de 4 ciclos de b, o valor de a é incrementado por NEXT a e, se não fosse o termo mencionado, o valor de a passaria bruscamente de 4 para 6 nesse ponto. Faça-se a experiência).

No ciclo «aleatório», o valor de b «salta» bruscamente ao exceder as várias condições «maior do que» (p. ex., IF b>8 THEN...). Finalmente, no ciclo de «escolha múltipla», é utilizada uma combinação de declarações lógicas e de salto condicionados para permitir que sejam extraídas sete sequências diferentes de números, de uma tabela de consulta neste caso concreto. Este processo é utilizado no programa «Cristais cúbicos» para determinar as permutações de faces de cristais tetragonais.

Estas sub-rotinas exemplificam operações de ciclo razoavelmente mais complexas do que as existentes em ciclos simples ou encaixados (*nested*).

24 CICLO SEPARADOR

```
1 2 3 4 13 14 15 16

FOR a=1 TO 5: IF a>4 THEN FOR
   a=13 TO 16
PRINT a;" ";
NEXT a
```

Este ciclo selecciona uma sequência de valores de a formada por dois grupos separados de 4 valores de a consecutivos:

1 2 3 4 13 14 15 16

25 CICLO MISTO

```
1,5 2,6 3,7 4,8 5,5 6,6 7,7 8,8


  FOR a=1 TO 7: FOR b=5 TO 8
  PRINT a;",";b;" ";
  LET a=a+1-(a=4): NEXT b: NEXT
    a
```

Este ciclo selecciona os 8 pares de valores das variáveis a e b seguintes:

1,5 2,6 3,7 4,8 5,5 6,6 7,7 8,8

26 CICLO «ALEATÓRIO»

```
FOR a=1 TO 4: FOR b=1 TO 2
IF b>1 THEN FOR b=7 TO 9
IF b>8 THEN FOR b=11 TO 15 ST
 EP 2
PRINT b;" ";: IF b<14 THEN NE
 XT b
PRINT " ";: NEXT a
```

Este ciclo imprime a sequência 1 7 8 11 13 15 quatro vezes.

27 CICLO DE ESCOLHA MÚLTIPLA

Esta sub-rotina selecciona sete grupos diferentes de números de um conjunto de 16 elementos, e é utilizada para escolher valores de uma tabela de consulta.


```
DIM T(16)
FOR a=1 TO 16: LET T(a)=a: NE
 XT a
FOR b=1 TO 7: FOR a=1 TO 5-(b
 =1) STEP 1+(b=5 OR b=7)
IF a>4 AND (b=2 OR b=5 OR b=5
 OR b=7) THEN FOR a=5 TO 9-(b
 =2 OR b=5) STEP 1+(b=6 OR b=
 7)
IF a>4+3*(b=6 OR b=5)+4*(b=7)
 AND (b=3 OR b=5 OR b=7) THEN
  FOR a=8+(b=3 OR b=7) TO 13-
 (b=3) STEP 1+(b=7)
IF a>4+7*(b=5 OR b=7) AND (b=
 4 OR b=5 OR b=7) THEN FOR a=
 12+(b=4 OR b=7) TO 16 STEP 1
 +(b=7)
PRINT T(a);",";: NEXT a
PRINT : NEXT b
```

Esta sub-rotina selecciona valores de a da seguinte forma:

b	*a*
1	1,2,3,4
2	1,2,3,4,5,6,7,8
3	1,2,3,4,9,10,11,12
4	1,2,3,4,13,14,15,16
5	1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16
6	1,3,5,7
7	1,3,5,7,9,11,13,15

28 INVERSÃO DE MATRIZES

Não podemos efectuar a divisão de duas matrizes, mas podemos calcular a matriz inversa de uma delas, a de A, por exemplo, representada por A^{-1}, e multiplicá-la pela outra matriz, empregando as regras para a multiplicação de matrizes, de modo a obtermos um resultado equivalente ao de uma divisão. No entanto, isto só é possível se a matriz A a inverter for quadrada e não singular (também designada por não degenerada, isto é, cujo determinante não seja nulo).

O cálculo da matriz inversa A^{-1} é baseado no método de Gauss-Jordan, tal como é dito no texto da sub-rotina 21, sendo portanto estabelecidos três ciclos que correspondem a cada uma das três operações que o método citado utiliza para cada um dos elementos individuais que compõem a matriz, os quais são tratados à vez, sendo, contudo, cada um dos seus valores na matriz imediatamente substituído pelo novo valor calculado. Assim, numa matriz de 3×3, são efectuadas 27 operações (3×9) de modo que a matriz transforma-se gradualmente na sua inversa em 27 passos. Será, talvez, bom relembrar que uma matriz, para ser invertível, tem de ser quadrada (número de linhas igual ao número de colunas) e não singular ou não degenerada, isto é, com o determinante respectivo diferente de zero.

A primeira linha da sub-rotina propriamente dita (quarta da listagem) verifica a existência ou não de um zero como valor do elemento do canto superior direito da matriz [elemento a(1,1), ou primeiro elemento da matriz] e, caso afirmativo, permuta as posições de duas linhas (sendo uma delas a que contém o elemento «incómodo») por meio da sub-rotina A, de modo a evitar que a rotina «estoire» na linha 2 (quinta da listagem) ao ser tentada uma divisão por zero. A linha B verifica se o resultado da subtracção da linha 6 é ou não um zero e faz a sub-rotina «abortar» (interromper a sua execução normal) caso seja, pois isso significa que o determinante da matriz é zero e que, portanto, esta não é invertível. A variável t, na primeira linha, fica com o seu valor alterado de zero para um no caso de haver necessidade

de trocar as posições de duas linhas, e depois permite a permuta das posições de duas colunas, na linha E, para compensar a troca de linhas. Repare-se na utilização de IF....THEN IF....THEN, na linha B, em vez de IF....AND....THEN, devido ao facto de, no caso de ser usada esta última declaração, apesar da primeira condição (IF i<>n) não ser verificada quando i=n, ambas as condições, isto é, ambas as partes da declaração AND serem testadas, o que levaria à emissão da mensagem de erro «Subscript wrong» por o termo A(n+1,n+1), resultante da condição i=n, não existir, como é óbvio. Assim, utilizando duas declarações IF, ao não ser verificada a condição da primeira, a sequência de execução da sub-rotina salta para a linha seguinte e a segunda já não é testada. Na linha 7, a utilização da declaração lógica −(i=n) evita a necessidade de empregar um segundo ciclo para lidar com os elementos da última linha da matriz, ciclo este que é encontrado em algumas versões desta rotina.

Esta sub-rotina inverte uma matriz A, dando como resultado a matriz inversa A^{-1}. Como se disse atrás, para que a inversão se possa efectuar é necessário que a matriz A seja quadrada e não singular, isto é, que o seu número de linhas seja igual ao seu número de colunas e que o seu determinante seja diferente de zero.

```
INPUT n: DIM A(n,n)
FOR i=1 TO n: FOR j=1 TO n
INPUT A(i,j): NEXT j: NEXT i

Utilizar, como alternativa, a
sub-rotina de validacao de input
15 (Matrizes e Quadros).

LET t=0: IF A(1,1)=0 THEN LET
  t=1: GO SUB A
FOR i=1 TO n: LET A(i,i)=1/A(
  i,i)
FOR j=1 TO n: IF j=i THEN GO
  TO B
LET A(j,i)=A(j,i)*A(i,i)
FOR k=1 TO n: IF k=i THEN GO
  TO C
```



```
LET A(j,k)=A(j,k)-A(j,i)*A(i,
  k)
IF j<>n-(i=n) THEN GO TO C
LET A(i,k)=-A(i,i)*A(i,k)
C NEXT k
B NEXT j: IF i<>n THEN IF ABS(A
  (i+1,i+1))<=1E-6 THEN LET t=2
  : GO TO R
NEXT i: GO TO E
A FOR p=2 TO n: IF A(p,1)<>0 TH
  EN GO TO F
NEXT p: GO TO D
F FOR j=1 TO n: LET z=A(p,j): L
  ET A(p,j)=A(1,j): LET A(1,j)
  =z: NEXT j
RETURN
E IF t THEN FOR j=1 TO n: LET z
  =A(j,p): LET A(j,p)=A(j,1):
  LET A(j,1)=z: NEXT j

R RETURN

R IF t<2 THEN FOR i=1 TO n: FOR
  j=1 TO n: PRINT "A(";i;",";
  j;")=";A(i,j): NEXT j: NEXT
  i: STOP
D PRINT "- Nao ha' solucao, poi
  s a matriz' degenerada (det
  erminante = 0)"

R Utilizar a subrotina 35,para
  impressao de matrizes ou de-
  terminantes
```

Exemplo

$$\begin{bmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \\ 1 & 0 & 3 \end{bmatrix} \longrightarrow \begin{bmatrix} -1.5 & 1.125 & 0.25 \\ 0 & 0.25 & -0.5 \\ 0.5 & -0.375 & 0.25 \end{bmatrix}$$

Nota: Para a impressão da matriz inversa acima indicada, foi utilizada a seguinte versão da sub-rotina 35:

```
IF t<2 THEN FOR i=1 TO n: FOR j
 =1 TO n: PRINT AT 2*i,8*j-7;A(
 i,j): NEXT j: NEXT i
PLOT 10,167: DRAW -8,0: DRAW 0,
 8-16*j: DRAW 8,0 -8,0: DRAW 0
PLOT 42*i,167: DRAW 8,0: DRAW 0
 ,8-16*j: DRAW -8,0
```

Nem sempre é possível apresentar o *output* de uma rotina destas na forma de um quadro ou matriz, pois os números, caso não sejam racionais, são impressos a oito algarismos significativos, o que torna extremamente difícil precisar a posição das chavetas e/ou dos diversos elementos da matriz, de modo a não haver algarismos que se sobreponham. Pode, contudo, utilizar-se uma das sub-rotinas de arredondamento (por exemplo, para quatro casas decimais) para superar este problema. Além disso, devido a imprecisões nos cálculos efectuados por esta sub-rotina, o valor zero de um elemento da matriz inversa é representado por um número muito pequeno, por exemplo, 1.3Ø16719E-9 em vez de Ø. Para evitar que isto aconteça, podemos acrescentar uma linha extra antes da declaração PRINT, tal como, por exemplo, a seguinte:

```
IF ABS A(i,j)<=1E-8 THEN LET A(i,j)=Ø
```


29 MULTIPLICAÇÃO DE MATRIZES

As matrizes são utilizadas num grande número de aplicações, nomeadamente no tratamento de quadros *(arrays)* de dados, como no exemplo apresentado aqui, na transformação de coordenadas quando se efectua uma mudança de sistema de eixos, em análises estatísticas de dados que envolvam cálculos de regressão, etc. As matrizes constituem, portanto, uma ferramenta matemática muito poderosa, mas também bastante complexa.

A multiplicação de duas matrizes é efectuada linha por coluna, e cada elemento c(j,k) da matriz produto é dado por:

$$c(j,k)=\sum_{i=1}^{n} a(j,i)*b(i,k)=a(j,1)*b(1,k)+a(j,2)*b(2,k)+\ldots$$

$$\ldots+a(j,n)*b(n,k)$$

utilizando uma notação semelhante à do BASIC, para maior clareza. Assim temos, por exemplo, que

$$\begin{bmatrix} a11 & a12 \\ a21 & a22 \end{bmatrix} * \begin{bmatrix} b11 & b12 \\ b21 & b22 \end{bmatrix} =$$

$$= \begin{bmatrix} a11*b11+a12*b21 & a11*b12+a12*b22 \\ a21*b11+a22*b21 & a21*b12+a22*b22 \end{bmatrix}$$

Na multiplicação de matrizes, a ordem de cada uma é muito importante, e uma matriz m*n só pode ser multiplicada por uma matriz n*r para que o resultado seja uma matriz m*r. Uma das consequências disto é ter-se, geralmente, que A*B seja diferente de B*A.

A presente sub-rotina multiplica uma matriz m*n por uma matriz n*r de modo a obter uma matriz-produto m*r.

```
INPUT "Numero de linhas de A
 ? ";m,"Numero de colunas de
 A? ";n,"Numero de colunas de
  B? ";r
DIM A(m,n): DIM B(n,r): DIM C
 (m,r)
```

```
Utilize a rotina de validacao de
input (matrizes) duas vezes para
introduzir A e B (veja a lista-
gem que acompanha o exemplo)


    FOR j=1 TO m: FOR i=1 TO r: L
     ET C(j,i)=0
    FOR k=1 TO n: LET C(j,i)=C(j,
     i)+A(j,k)*B(k,i)
    NEXT k: NEXT i: NEXT j
R   RETURN


R   CLS : PRINT AT 1,8;"A MATRIZ
     PRODUTO E":": FOR j=1 TO 20:
     POKE 16455+j,255: NEXT j
    FOR j=1 TO m: FOR k=1 TO r
    PRINT AT 2*j+3,4*k-3;C(j,k)
    NEXT k: NEXT j
    PLOT 10,140: DRAW -8,0: DRAW
     0,16-16*j: DRAW 8,0
    PLOT 32*k-38,140: DRAW 8,0: D
     RAW 0,16-16*j: DRAW -8,0
```

Exemplo

Se considerarmos que a seguinte tabela dá o número de funcionários, por letras de vencimento, em seis empresas públicas,

Empresa ∨	G	F	E	D	C	< Letra
1	200	50	10	1	0	
2	250	60	6	2	0	
3	500	101	12	3	0	
4	35	5	1	0	1	
5	1010	190	27	2	1	
6	1250	200	45	3	1	

e que os respectivos vencimentos anuais, com 13.º mês, são

Letra	Contos/ano
G	577
F	605
E	654
D	735
C	852

então,

$$\begin{bmatrix} 200 & 50 & 10 & 1 & 0 \\ 250 & 60 & 6 & 2 & 0 \\ 500 & 101 & 12 & 3 & 0 \\ 35 & 5 & 1 & 0 & 1 \\ 1010 & 190 & 27 & 2 & 1 \\ 1250 & 200 & 45 & 3 & 1 \end{bmatrix} * \begin{bmatrix} 577 \\ 605 \\ 654 \\ 735 \\ 852 \end{bmatrix} = \begin{bmatrix} 152925 \\ 185944 \\ 359658 \\ 24726 \\ 717700 \\ 874737 \end{bmatrix}$$

cuja representação simbólica é

$$a(m,n)*b(n,r)=c(m,r)$$

dará a verba total anual despendida em vencimentos por cada uma das empresas públicas. Os dados relativos aos vencimentos foram calculados com base nos vencimentos da função pública para 1985. Não faria muito sentido calcular o exemplo para um grupo heterogéneo de empresas privadas e públicas (a não ser que fizessem todas parte de um mesmo grupo financeiro) porque as tabelas salariais apresentariam, em tal caso, flutuações consideráveis, o que iria introduzir uma complexidade desnecessária a um simples exemplo. No original inglês, o processo utilizado é semelhante.

Nota: Devido às dificuldades que podem ser encontradas ao incluir a sub-rotina de validação de *input* para matrizes (n.º 15) nesta sub-rotina, de modo a permitir uma fácil introdução dos

elementos das duas matrizes a multiplicar (lembremo-nos de que isto vai obrigar a que a dita sub-rotina de *input* seja utilizada duas vezes), apresentamos seguidamente, a título de exemplo, uma listagem completa que inclui as duas sub-rotinas mencionadas devidamente interligadas. Repare-se na utilização da variável oo, destinada a permitir que as duas matrizes sejam introduzidas separadamente sem que a rotina de *input* tenha de ser escrita duas vezes. Na linha 27Ø, a declaração PLOT 32*k−1Ø está calculada de modo a que a chaveta direita da matriz-produto do exemplo apresentado seja posicionada correctamente. Noutros casos, será eventualmente necessário alterar os valores incluídos nessa declaração. Ver, a este propósito, a nota que acompanha a rotina anterior (n.º 28).

A matriz A é:

```
[200 50  10  1   Ø
250 60   6   2   Ø
500 101  12  3   Ø
35  5    1   Ø   1
1010190  27  2   1
1250200  45  3   1 ]
```

A matriz B é:

```
[577
605
654
735
852]
```


A MATRIZ PRODUTO É:

```
[152925
185944
359658
24726
717700
874737]
```

```
   1 REM Multiplicacao
       de Matrizes
   2 INPUT "Numero de linhas de
A ? ";m,"Numero de colunas de A?
 ";n,"Numero de colunas de B? ";
r
   3 DIM A(m,n): DIM B(n,r): DIM
 C(m,r)
   5 LET oo=Ø: LET s=m: LET t=n
  10 IF oo=1 THEN LET s=n: LET t
=r: CLS
  20 PRINT AT 1,8;"A matriz ";"A
" AND (oo=Ø);"B" AND (oo=1);"é
:"
  30 FOR j=1 TO 14: POKE 16455+j
,255: NEXT j
  40 DIM M(s,t): LET k$="s"
  50 FOR j=1 TO s: FOR k=1 TO t
  60 PRINT AT 3+2*j,4*k-3;"a" AN
D (oo=Ø);"b" AND (oo=1);+STR$ j+
STR$ k
  70 NEXT k: NEXT j
  80 PLOT 10,140: DRAW -8,Ø: DRA
W Ø,16-16*j: DRAW 8,Ø
  90 PLOT 32*k-38,140: DRAW 8,Ø:
 DRAW Ø,16-16*j: DRAW -8,Ø
 100 FOR j=1 TO s: FOR k=1 TO t
 110 INPUT "Introduza os valores
 numericos  ";"outra vez, correc
tamente" AND (k$<>"s");;"a" AND
 (oo=Ø);"b" AND (oo=1);+STR$ j+S
TR$ k;" = ";M(j,k)
```

```
 120 PRINT AT 2*j+3,4*k-3;"    ";
CHR$ 8;CHR$ 8;CHR$ 8;M(j,k)
 130 NEXT k: NEXT j
 140 INPUT "              OK? (s/n) "
;k$
 150 IF k$<>"s" THEN GO TO 100
 160 IF oo=0 THEN FOR j=1 TO m:
FOR k=1 TO n: LET A(j,k)=M(j,k):
 NEXT k: NEXT j: LET oo=1: GO TO
 10
 170 FOR j=1 TO n: FOR k=1 TO r:
 LET B(j,k)=M(j,k): NEXT k: NEXT
 j
 190 FOR j=1 TO m: FOR i=1 TO r:
 LET C(j,i)=0
 200 FOR k=1 TO n: LET C(j,i)=C(
j,i)+A(j,k)*B(k,i)
 210 NEXT k: NEXT i: NEXT j
 220 CLS : PRINT AT 1,8;"A MATRI
Z PRODUTO E':": FOR j=1 TO 20: P
OKE 16455+j,255: NEXT j
 230 FOR j=1 TO m: FOR k=1 TO r
 240 PRINT AT 2*j+3,4*k-3;C(j,k)
 250 NEXT k: NEXT j
 260 PLOT 10,140: DRAW -8,0: DRA
W 0,16-16*j: DRAW 8,0
 270 PLOT 32*k-10,140: DRAW 8,0:
 DRAW 0,16-16*j: DRAW -8,0
```


30 MÍNIMO, MÁXIMO, MÉDIA, MEDIANA E MODA

Se, dado um conjunto de números, os colocarmos em ordem ascendente, é fácil determinar o mínimo e o máximo desse conjunto (menor e maior valor, respectivamente). A média é determinada a partir da soma S de todos os valores, calculada na fase de introdução destes. Achado o valor da soma, este é dividido pelo número, n, de valores introduzidos para se obter a média. A mediana é outra «medida da tendência central» e é numericamente igual ao valor central (se este existir, ou seja, quando n for ímpar) ou a média aritmética dos dois valores centrais (quando n for par). Entenda-se por valor central o valor que se encontra exactamente a «meio» do conjunto de números, estando este ordenado, ou seja, é o valor de ordem n/2, no caso de n ser par. Caso contrário, existirão dois «valores centrais».

A moda é o valor mais difícil de determinar. A moda de um conjunto de números é o valor que ocorre com maior frequência nesse conjunto, isto é, é o valor mais comum. A moda pode não existir (se os valores ocorrerem todos com a mesma frequência) ou não ser única. Desta forma, para efectuar o cálculo da moda, a sub-rotina tem de contar o número de vezes que um mesmo valor é introduzido, para determinar se algum (ou alguns) é introduzido mais vezes do que os outros. Esta contagem é efectuada por T(1,z) na linha C. Esta linha faz parte de um pequeno ciclo que vai percorrendo todos os valores V(p) do conjunto de números e contando as vezes que tal ou tal valor é repetido. T(2,z), na linha seguinte, armazena os vários valores. A linha E ordena então os valores T(1,z) por ordem descrescente e a moda corresponderá assim ao primeiro valor em T(2,z). A linha D efectua as trocas de posição necessárias à ordenação, alterando, se necessário, tanto as posições de T(1,z) como as de T(2,z). O processo utilizado nesta ordenação é o descrito na sub-rotina 5 (Ordenação).

É claro que o método de cálculo da moda descrito acima tem os seus inconvenientes, a par de uma relativa simplicidade. O mais flagrante é a incapacidade de lidar com casos em que não existe

moda (todos os valores apresentam a mesma frequência) ou em que haja mais do que uma moda. Em ambos os casos, a sub-rotina apresenta um valor (único) para a moda, como se ela existisse ou fosse uma só. No caso de todos os valores terem a mesma frequência, é apresentado como moda o valor mínimo (visto ser o último na ordenação por ordem decrescente), enquanto, no caso de haver mais do que uma moda, o valor apresentado é o menor dos que apresentam a frequência máxima (ou modal), visto ser o último na ordenação decrescente, não sendo apresentados os restantes valores da moda. É claro que se ultrapassa este inconveniente com a inclusão de uma sub-rotina suplementar que lide com estes casos (ou com a alteração da sub-rotina existente). No entanto, tal inclusão ou alteração iria dar uma certa complexidade à sub-rotina e um aumento significativo do tempo de execução, que é já de certo modo importante para grandes conjuntos de números (cerca de um minuto para 50 números).

Assim, esta sub-rotina imprime os valores mínimos, máximo, da média, da mediana e da moda de um conjunto de números positivos.

```
    INPUT "Numero de valores? ";n
     : DIM V(n)
    LET s=0: FOR p=1 TO n: INPUT
     "V(";(p);")=";V(p): LET s=s+
     V(p): NEXT p

Utilizar, como alternativa, a
sub-rotina 15 para validacao de
input para introduzir n e V(n).


B   LET q=0: FOR p=1 TO n-1
    IF V(p+1)<V(p) THEN GO SUB A:
     LET q=q+1
    NEXT p: IF q<>0 THEN GO TO B
    LET p=1: LET z=1: DIM T(2,n)
C   IF V(p)=V(p+1) THEN LET T(1,z
     )=T(1,z)+1: LET p=p+1: IF p<
     n THEN GO TO C
    LET T(2,z)=V(p): LET p=p+1: L
     ET z=z+1+T(1,z): IF p<n THEN
      GO TO C
E   LET q=0: FOR p=1 TO n-1: IF T
     (1,p+1)>T(1,p) THEN GO SUB D
     : LET q=q+1
    NEXT p: IF q<>0 THEN GO TO E
    PRINT "Minimo = ";V(1);"Maxi
     mo = ";V(n);"Media  = ";s/n
    IF 2*INT (n/2)=n THEN PRINT "
     Mediana= ";(V(n/2)+V(n/2+1))
     /2: GO TO F
    PRINT "Mediana= ";V(n/2)
F   PRINT "Moda   = ";T(2,1)
    GO TO R
A   LET a=V(p): LET V(p)=V(p+1):
     LET V(p+1)=a: RETURN
D   LET a=T(1,p): LET b=T(2,p): L
     ET T(1,p)=T(1,p+1): LET T(2,
     p)=T(2,p+1): LET T(1,p+1)=a:
     LET T(2,p+1)=b: RETURN
R   RETURN
```

31 FILTRO DE NOMES

Esta sub-rotina «filtra» nomes (próprios ou não, substantivos, etc.), rejeitando todos aqueles que apresentem caracteres que não sejam normais em nomes, tais como algarismos, cifrões, sinais de ponto e vírgula, de dois pontos, etc. Assim, é possível detectar alguns tipos de erros que ocorrem frequentemente ao digitar listas de nomes (de titulares de contas bancárias e outras), evitando que estes vão causar problemas em programas que lidem com ficheiros que incluam esses nomes, etc.

A sub-rotina permite o armazenamento dos vários nomes a controlar. Os caracteres aceitáveis nesses nomes podem ser letras maiúsculas ou minúsculas, traços de união, pontos, espaços em branco e apóstrofos. As palavras que contenham qualquer outro tipo de caracteres são rejeitadas.

Na primeira linha da sub-rotina, o número 20 na declaração DIM B$(n,2Ø) permite, em princípio, dimensionar o quadro alfanumérico (ou de cadeias) B$ de modo a que este possa conter a maioria dos nomes mais comuns, utilizando, se necessário, iniciais para reduzir o comprimento de alguns deles. Contudo, esse valor pode ser alterado de modo a que o quadro possa acomodar nomes de maior comprimento (ou menor, caso a grande maioria dos nomes de um grupo tenha um comprimento total nitidamente inferior, o que iria desperdiçar espaço de memória se o valor não fosse alterado).

O ciclo controlado por p testa os nomes carácter a carácter, à medida que aqueles vão sendo introduzidos, para verificar se todos os caracteres que os constituem são ou não aceitáveis. Caso algum nome não «passe» no teste, o valor de q é diminuído de 1 de modo a que não seja deixada uma linha de 20 espaços em branco em B$(n,20).

```
INPUT "Numero de nomes? ";n:
 DIM B$(n,20)
```

```
    FOR q=1 TO n: INPUT "Nome ";(
     q);" : ";A$
    LET B$(q)=A$
    FOR p=1 TO LEN A$
    IF A$(p)="-" OR A$(p)="," OR
     A$(p)=" " OR A$(p)="" OR A$(
     p)>="A" AND A$(p)<="Z" OR A$
     (p)>="a" AND A$(p)<="z" THEN
      NEXT p: GO TO A
    PRINT A$;" nao e'um nome": LE
     T q=q-1
A   NEXT q
R   RETURN

R   PRINT : FOR q=1 TO n: PRINT B
     $(q): NEXT q
```

Exemplo

```
 Numero de nomes? 12

 Yoko Tsuno nao e'um nome
 N, Maquiavel nao e'um nome

 Verbo
 I. Sinclair
 Sigma Press
 Yoko Tsuno
 R. Wagner
 N. Bonaparte
 Albert Einstein
 Piotr I. Tchaikowsky
 J. Brahms
 N. Maquiavel
 Sr. Silva
 Alvaro de Campos
```


32 PERMUTAÇÕES DE TRÊS NÚMEROS

Esta sub-rotina gera as seis permutações possíveis de três números h, k e l quaisquer.

Como o primeiro número pode ser escolhido de três maneiras diferentes, o segundo apenas de duas (pois já escolhemos um dos números para o primeiro) ficando então o terceiro determinado (só pode ser escolhido de uma única maneira), o número total de maneiras diferentes pelas quais podemos dispor (permutar) os três números é 3*2*1=6.

De um modo geral, podemos dizer que o número total de permutações de n objectos diferentes (números, etc.) é dado por

$$Pn=n!$$

onde Pn designa o «número de permutações de n» e n! é «n factorial» (ver sub-rotina 22).

Nesta sub-rotina, a escolha começa no ciclo comandado por a com

h k l

Para que a sequência de permutações seja elaborada, é trocada, em cada execução do ciclo, a posição relativa de dois dos números, dependendo esta troca do valor de t, que é 3 (quando a é ímpar) ou 2 (quando a é par) e, assim, temos:

a=1	t=3	h k l	trocar	1 e 3
a=2	t=2	l k h	trocar	1 e 2
a=3	t=3	k l h	trocar	1 e 3
a=4	t=2	h l k	trocar	1 e 2
a=5	t=3	l h k	trocar	1 e 3
a=6	t=2	k h l	trocar	1 e 2
		h k l		

```
h=1  k=2  l=3

1  2  3
3  2  1
2  3  1
1  3  2
3  1  2
2  1  3
```

```
DIM A(3): INPUT "h=";A(1);"
  k=";A(2);"  l=";A(3)

DIM P(6,3): FOR a=1 TO 6: LET
  t=2+2*(a/2-INT (a/2))
FOR b=1 TO 3: LET P(a,b)=A(b)
  : NEXT b
LET C=A(1): LET A(1)=A(t): LE
  T A(t)=C: NEXT a
R RETURN

R FOR a=1 TO 6: FOR b=1 TO 3
  PRINT P(a,b);"  ";: NEXT b
  PRINT : PRINT : NEXT a
```


33 INFLAÇÃO

Esta sub-rotina mostra a desvalorização do dinheiro em Portugal entre 1967 e 1984 e a influência dessa desvalorização nos preços no consumidor.

Os dados necessários à elaboração desta sub-rotina foram calculados com base nos índices de preços no consumidor para os anos do intervalo indicado acima. Contudo, convém referir que, devido a uma quebra na série desses índices, em 1976, por estes passarem a ser calculados para todo o Continente (anteriormente eram apenas para Lisboa), poderão existir algumas pequenas inexactidões nos cálculos efectuados pela sub-rotina, principalmente se uma das datas for anterior a 1976 e a outra posterior. Todos os índices consultados excluíam a habitação.

```
DATA 1.00,.993,.9643,.9434,.9
 05,.8658,.8123,.6738,.5054,.
 5115,.4568,.4217,.3885,.3702
 ,.3494,.3282,.3065,.2843
DIM I(18): FOR i=1 TO 18: REA
 D I(i): NEXT i

INPUT "Introduza o valor e o
 ano de      compra (1967-1984)
  : ";V;"$00 em 19";y
B PRINT "Para que ano quer sabe
 r o valor"
INPUT "19";a
IF a>=67 AND a<=84 THEN GO TO
 A
PRINT "Nao existem dados para
  esse ano": GO TO B
A PRINT : PRINT "Custou ";V;"$0
 0 em 19";y
LET VV=V*I(a-66)/I(y-66): LET
  VA=V*I(y-66)/I(a-66)
PRINT : PRINT "O valor equiva
 lente dessa      quantia em
 19";a;" seria de";INT VV;"
$";INT ((VV-INT VV)*10);"0"
;"Custaria pois ";INT VA;"$"
;INT ((VA-INT VA)*10);"0";"s
```

```
e fosse comprado em 19";a/"
O factor e'de ";I(y-66)/I(a-
66);" para 1."

R   RETURN
```


34 NÚMEROS PRIMOS

Como sabemos, um número primo é um inteiro cuja divisão exacta (resto zero) só é possível se o divisor for a unidade ou o próprio número.

Estas duas sub-rotinas geram uma lista de números primos desde um até um limite especificado. Ambas tomam os números 1, 2 e 3 como primos sem efectuarem quaisquer cálculos. A partir de 3, é a variável a que vai representar cada número primo da lista. Esta variável é incrementada sucessivamente em 2, de forma a tomar sempre valores ímpares (lembremo-nos de que um número primo é sempre ímpar, pois qualquer número par é sempre divisível por 2). É utilizada uma segunda variável, b, que também vai assumindo apenas valores ímpares e que é inicializada em 3 para cada valor de a. Cada um desses valores de a é dividido pelos sucessivos valores de b, e cada divisão é testada para se ver se é ou não exacta (se o resto é zero ou diferente de zero). Se existir um valor de b para o qual a divisão seja exacta, então o valor de a não é primo e é, consequentemente, incrementado em 2 e testado novamente. Se nenhum dos valores de b for divisor exacto de a, o valor de a é primo e é impresso pela linha B antes de ser incrementado para o valor seguinte. As divisões por b vão sendo efectuadas até o valor de b atingir ou exceder um valor aproximadamente igual ao da raiz quadrada do número (valor de a) a ser testado (ver sétima linha das listagens). Isto deve-se ao facto de não haver nenhum divisor inteiro superior à raiz quadrada do número desde que não se tenha encontrado algum inferior a esse valor. Seria, portanto, inútil continuar a incrementar o valor do divisor b e a efectuar a divisão de a.

A segunda sub-rotina difere da primeira em alguns pontos, nomeadamente na definição do valor-limite para a (SQR n, ou seja, raiz quadrada de n, em vez de n) e na capacidade de efectuar uma contagem dos números primos gerados. Essa contagem é

conseguida efectuando uma contagem do número de elementos de N(p) diferentes de zero, no ciclo de impressão R.

Sub-rotina 34.1
Esta sub-rotina gera uma lista dos números primos existentes até um número N.

```
      INPUT N

      LET n=ABS N: LET a=3
      IF n>=1 THEN PRINT 1
      IF n>=2 THEN PRINT 2
      IF n>=3 THEN PRINT a
A     LET b=3: LET a=a+2
C     IF a-b*INT (a/b)=0 THEN GO TO
       A
      IF b>=INT (a/b) THEN GO TO B
      LET b=b+2: GO TO C
B     IF a<=n THEN PRINT a
      IF a>=n THEN GO TO R
      GO TO A
R     RETURN
```

Sub-rotina 34.2
Esta sub-rotina gera uma lista dos números primos até SQR N, armazena-os no quadro N() e calcula o seu número.

```
      INPUT N

      LET n=ABS N: LET a=3: DIM N(S
       QR n/2+2): LET z=4
      IF n>=1 THEN LET N(1)=1
      IF n>=4 THEN LET N(2)=2
      IF n>=9 THEN LET N(3)=3
A     LET b=3: LET a=a+2
C     IF a-b*INT (a/b)=0 THEN GO TO
       A
      IF b>=INT (a/b) THEN GO TO B
      LET b=b+2: GO TO C
B     IF a<=SQR n THEN LET N(z)=a:
       LET z=z+1
      IF a>=SQR n THEN GO TO R
      GO TO A
R     RETURN
```



```
R     FOR p=1 TO SQR n/2+2: IF N(p)
       <>0 THEN PRINT N(p)
      NEXT p
      PRINT "*O numero de numeros p
       rimos ate' SQR ";n;" (";SQR
       n;") e' ";p-1
```

35 APRESENTAÇÃO GRÁFICA DE MATRIZES E DETERMINANTES

Como se disse na nota que acompanha a sub-rotina 28, nem sempre é muito fácil apresentar graficamente uma matriz na sua forma «matricial» clássica, devido às diferenças de comprimento (em número de dígitos) que os elementos, principalmente os não inteiros, podem apresentar. Assim, para utilizar esta sub-rotina, pode ser necessário efectuar um arredondamento (por meio de uma das sub-rotinas de arredondamento) dos vários elementos da matriz ou determinante para quatro algarismos significativos, se eles apresentarem valores não inteiros.

Esta sub-rotina imprime quadros com um máximo de 9 linhas e 7 colunas, com «chavetas» de matriz ou de determinante.

```
A   INPUT "Numero de linhas? ";m,
     "Numero de colunas? ";n
    IF m>9 OR n>7 THEN GO TO A

    DIM A(m,n): FOR j=1 TO m: FOR
     k=1 TO n
    INPUT A(j,k): NEXT k: NEXT j

    FOR j=1 TO m: FOR k=1 TO n
    PRINT AT 2*j,4*k-3;A(j,k)
    NEXT k: NEXT j
    PLOT 10,167: DRAW -8,0: DRAW
     0,8-16*j: DRAW 8,0
    PLOT 32*k-30,167: DRAW 8,0: D
     RAW 0,8-16*j: DRAW -8,0
R   RETURN

    Para determinantes utilize

    PLOT 2,167: DRAW 0,8-16*j
    PLOT 32*k-22,167: DRAW 0,8-16
     *j
```

Exemplo

```
Matriz

| 375 519 23  812 159 123 620 |
| 200 136 179 66  17  541 32  |
| 0   51  23  17  2   33  5.5 |
| 615 431 212 191 122 166 32  |
| 400 317 619 221 166 7.5 31  |

Determinante

| 7   5   6   7   0  |
| 15  25  36  7   14 |
| 22  44  14  20  13 |
| 42  34  61  78  13 |
| 0   32  17  23  10 |
```

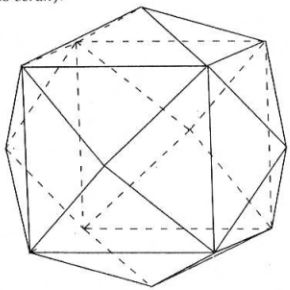

36 PROJECÇÃO

Trata-se aqui de uma projecção paralela sobre um plano, e a sua utilização é evidenciada nas várias figuras representando formas cristalinas.

A transformação pode ser expressa, segundo notação matricial, da seguinte forma:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} -\cos\alpha & \cos\beta & \emptyset \\ -\text{sen}\alpha & -\text{sen}\beta & 1 \\ \emptyset & \emptyset & \emptyset \end{bmatrix} * \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} 128 \\ 87 \\ \emptyset \end{bmatrix}$$

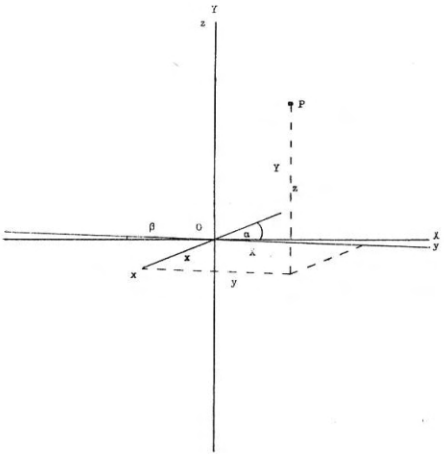
Trata-se, pois, de uma transformação de coordenadas, de um sistema de coordenadas x, y, z, para um novo sistema de coordenadas X, Y, Z, sendo o plano considerado o do *écran*, pertencente ao sistema X, Y, Z, e tendo como eixos coordenados X e Y. Os números 128 e 87 dão a deslocação da origem das coordenadas de $\emptyset,\emptyset$ para o ponto de coordenadas 128,87 do *écran* (centro do *écran*).



Estrutura cristalina

O diagrama seguinte mostra a relação existente entre um ponto P de coordenadas espaciais (tridimensionais) x,y,z e a sua projecção no plano X,Y que pode, por exemplo, ser o do *écran*.

α=arctan 1/3
β=arctan 1/27
O é a origem dos eixos e tem as coordenadas 128,87 no *écran*
Oz=72
Oy=72.05
Ox=25.3



A projecção do ponto P (x,y,z) é o ponto P(X,Y), onde

```
X=y cosβ-xcosα+128=Y*72.05*COS ATN (1/27)-x*25.3*COS ATN (1/3)+128
Y=z-x senα-y senβ+87=z*72-X*25.3*SIN ATN (1/3)-Y*72.05*SIN ATN (1/27)+87
```

Na última parte de cada igualdade figura a expressão equivalente em BASIC para o *Spectrum*.

As coordenadas x,y,z têm origem em O.

As coordenadas X,Y têm origem no canto inferior esquerdo do *écran* (com coordenadas $\emptyset$,$\emptyset$).

37 NÚMEROS INTEIROS PITAGÓRICOS

Esta sub-rotina determina pares de números, menores do que um dado valor, cujos quadrados, quando somados, dão um quadrado perfeito (por exemplo, 3 e 4, cujos quadrados, 9 e 16 dão, quando somados, 25, que é um quadrado perfeito). A sub-rotina exemplifica também uma forma de superar imprecisões nos cálculos efectuados pelo computador.

Para evitar que um par de números seja apresentado duas vezes, por exemplo, 3,4 e 4,3, o ciclo controlado pela variável b é inicializado em a. O outro ponto a notar é a necessidade de adicionar a t um valor extremamente pequeno para cobrir as já referidas inexactidões nos cálculos efectuados pelo *Spectrum*. De facto, apesar de um número poder aparecer impresso no *écran* como um inteiro, ao ser efectuada a comparação na quinta linha da sub-rotina, ele pode ser representado internamente em binário como um número não inteiro (.9999999 em vez de 1, por exemplo) e ser arredondado pela função INT, ficando obviamente t diferente de INT t, e a diferença comparativa resultar (erradamente) diferente de zero.

De referir ainda que a execução desta sub-rotina é bastante lenta, levando cada par de números cerca de 20 segundos a ser determinado, e, assim, o tempo total de execução para n=30, por exemplo, é de cerca de 4 minutos.

```
      INPUT "Valor maximo a ser con
       siderado: ";n

      FOR a=1 TO n
      FOR b=a TO n
      LET t=SQR (a↑2+b↑2)
      IF ABS (t-INT (t+.00000005))<
       =.00000005 THEN PRINT TAB 0;
       a;TAB 20;t;TAB 10;b
      NEXT b: NEXT a
R     RETURN
```

38 EQUAD

Esta sub-rotina calcula as raízes, reais ou imaginárias, de uma equação do 2.º grau (equação quadrática) dada na sua forma canónica, $ax^2+bx+c=\emptyset$, com base nas fórmulas resolventes seguintes:

$$x_1 = \frac{-b+\sqrt{b^2-4ac}}{2a} \qquad x_2 = \frac{-b-\sqrt{b^2-4ac}}{2a}$$

A estrutura da sub-rotina é bastante simples e clara, não obstante ser à prova de erros de *input* (o pedido de *input* é repetido se os valores introduzidos levarem a uma impossibilidade de resolução por a equação se tornar numa condição impossível). No caso de a ser zero (e b $\emptyset$), a equação transforma-se numa equação linear a uma incógnita e, como tal, só terá uma raiz, não utilizando a sub-rotina, nesse caso, as «fórmulas resolventes», mas sim a simples passagem de uma parcela (c) e de um factor numérico (b) de um membro para o outro, respectivamente, com mudança de sinal e inversão (terceira linha da listagem). No caso de b^2-4ac ser um número negativo, as raízes são imaginárias e a sub-rotina imprime-as na sua forma complexa p+r*i, sendo $r=\sqrt{q}$.

Repare-se que era possível encadear r$, «+» e i$ ou r$, «−» e i$ para a impressão das raízes imaginárias; contudo, isto não funcionaria se o valor de q se situasse entre $\emptyset$ e 1. Experimente-se introduzir o seguinte, em modo directo (sem número de linha):

```
PRINT "a" +STR$ 3, "a"; STR$ 3
```

e

```
PRINT "a" +STR$ .3, "a"; STR$ .3
```

É esta a razão pela qual as raízes imaginárias são impressas do modo utilizado na sub-rotina, ou seja, utilizando separadores em

vez de constituírem uma única cadeia formada pelo encadeamento de várias cadeias (*strings*).

```
A   INPUT "a=";a;" b=";b;" c=";c
    IF a=0 AND b=0 THEN GO TO A
    IF a=0 THEN LET b$="Unica": L
     ET x=-c/b: GO TO R
    LET p=-b/2/a: LET q=(b*b-4*a*
     c)/4/a/a
    IF q>=0 THEN LET b$="reais":
     LET x1=p+SQR q: LET x2=p-SQR
      q: GO TO R
    LET b$="imaginarias": LET r$=
     STR$ p: LET i$=STR$ SQR ABS
     q+"*i"
R   RETURN

R   IF b$(1)="r" OR b$(1)="i" THE
     N LET k$="As raizes ": LET l
     $=" sao: "
    IF b$(1)="u" THEN LET k$="A r
     aiz ": LET l$=" e' : "
    PRINT k$;b$;l$
    IF b$(1)="u" THEN PRINT %x
    IF b$(1)="r" THEN PRINT %x1,x
     2
    IF b$(1)="i" THEN PRINT %r$;"
     +";i$,r$;"-";i$
```


39-43 ARREDONDAMENTO DE NÚMEROS

As rotinas 42 e 43 são bastante úteis quando se pretende controlar a precisão de uma resposta numérica impressa, particularmente quando a impressão é efectuada em áreas limitadas do *écran*. Na rotina 42, o caso trivial de n=Ø é tratado em primeiro lugar. Seguidamente, as linhas 2 e 3 (LET B$... e FOR p...) fornecem uma série de zeros para preenchimento de casas decimais, se necessário. Na linha imediatamente a seguir é localizado o ponto decimal (vírgula) mas, caso não haja um (se o número for inteiro), o ponto é acrescentado, seguido de zeros, tantos quantos necessários para preencher as n casas decimais. O acréscimo de zeros para perfazer o número n de casas decimais pretendidas é também efectuado no caso de existir o ponto decimal e o número de casas decimais do número em questão ser inferior a n, sendo esse acréscimo efectuado pela linha 6. No caso de o número de casas decimais ser superior a n, o número a ser tratado é truncado a n casas decimais na linha 7.

Na sub-rotina 43 existem também as instruções necessárias ao fornecimento de uma série de zeros para preenchimento de casas. A posição do ponto decimal é determinada pela linha 4 mas, no caso de este não existir, a execução da rotina não salta para a linha A e a linha 5 é executada. É esta a linha que acrescenta então zeros ao número, conforme necessário.

A linha A desta sub-rotina trata do caso de o ponto decimal se situar no início do número (número sem parte inteira) e a sub-rotina B determina onde começam os dígitos significativos (posição esta dada por q), sendo estes truncados, ou não, como necessário.

Por fim, se o número n for composto de parte inteira e parte não inteira (fraccionária), a truncagem é efectuada deslocando a posição do ponto decimal, acrescentando Ø.5, tomando a parte inteira (com INT) e deslocando para trás o ponto.

Além de arredondarem um número para um inteiro com a função INT x, estas sub-rotinas permitem arredondar números de várias outras maneiras.

39 ARREDONDAR PARA UM INTEIRO, POR EXCESSO

```
      INPUT x

      IF x=INT x THEN GO TO R
      LET x=1+INT x
R     RETURN

R     PRINT x
```

40 ARREDONDAR PARA O INTEIRO MAIS PRÓXIMO

```
      INPUT x

      LET x=INT (x+0.5)
R     RETURN

R     PRINT x
```

41 CONVERSÃO DE HORAS

Nota: A sub-rotina 41 não é propriamente uma sub-rotina de arredondamento de números, embora inclua uma variação da sub-rotina 42. É uma sub-rotina de conversão de horas, dadas como um número decimal, com parte inteira e parte não inteira, em horas, minutos e segundos e vice-versa. A conversão de horas, minutos e segundos para horas, como número decimal, permite especificar o número de casas decimais com que se deseja que esse número seja apresentado, podendo-se assim imprimi-lo numa área delimitada do *écran*. Por outro lado, quanto à conversão inversa, bastará uma fácil alteração das declarações PRINT do corpo principal da sub-rotina para se obter uma apresentação gráfica do *output* algo diferente, como por exemplo, 23:Ø6:12. Neste caso, o número de caracteres impressos poderia ser sempre constante ou, quando muito, ter um valor máximo.

A sub-rotina não apresenta quaisquer dificuldades, sendo de notar o emprego da função AND nas declarações PRINT mencionadas, para permitir a impressão dos casos singular e plural, considerando o Ø plural (p. ex. Ø horas). A sub-rotina E é a sub-rotina 42 com pequenas alterações.

```
PRINT "Converter:"/"Horas -
 horas/minutos/segundos:1"/"
 Horas/minutos/segundos - hor
 as:2": REM Se o numero de ho
 ras > 24 a primeira conversa
 o inclui tambem dias.
INPUT "Introduza a opcao dese
 jada: ";k
IF k=2 THEN GO TO B


CLS : INPUT "Introduza t em h
 oras: ";t
LET h=INT t: IF t<24 THEN GO
 TO A
LET d=INT (t/24): LET h=INT (
 (t/24-d)*24)
```

```
A  LET m1=(t-INT t)*60: LET m=IN
   T m1
   LET s=INT ((m1-INT m1)*60)
   GO TO C
B  CLS : INPUT "Horas: ";h,"Minu
   tos: ";m,"Segundos: ";s
   LET h1=(s/60+m)/60
   LET t1=h+h1: GO SUB E
   PRINT h;" hora";"s" AND (h<>1
   )/m;" minuto";"s" AND (m<>1)
   ;" e";s;" segundo";"s" AND (
   s<>1)//"= ";A$;" horas"
   GO TO R
C  IF t>=24 THEN GO TO D
   PRINT t;" horas ="//h;" hora"
   ;"s" AND (h<>1)/m;" minuto";
   "s" AND (m<>1)/s;" segundo";
   "s" AND (s<>1)
   GO TO R
D  PRINT t;" horas ="//d;" dia";
   "s" AND (t>=48)/h;" hora";"s
   " AND (h<>1)/m;" minuto";"s"
    AND (m<>1)/s;" segundo";"s"
    AND (s<>1)
   GO TO R


E  INPUT "Numero de casas decima
   is? ";n
   LET A$=STR$ t1
   IF n=0 THEN LET A$=STR$ INT (
   t1+.5): GO TO G
   LET B$="": LET C$="0": LET l=
   LEN A$
   FOR p=1 TO n: LET B$=B$+C$: N
   EXT P
   FOR p=1 TO l: IF A$(p)="." TH
   EN GO TO F
   NEXT p: LET A$=A$+".":B$: GO
    TO G
F  IF n+p>=l THEN LET A$=A$+B$(
   TO n+p-l): GO TO G
   LET A$=A$( TO p+n-1)+STR$ INT
     (VAL A$(p+n TO )/10^(l-n-p)
   +.5)
G  RETURN


R  RETURN
```


42 ARREDONDAR PARA N CASAS DECIMAIS

```
   INPUT "Introduza o numero :";
    LINE A$: INPUT "Numero de c
   asas decimais? ";n


   IF n=0 THEN PRINT INT (VAL A$
   +.5): GO TO R
   LET B$="": LET C$="0": LET l=
   LEN A$
   FOR p=1 TO n: LET B$=B$+C$: N
   EXT P
   FOR p=1 TO l: IF A$(p)="." TH
   EN GO TO A
   NEXT p: PRINT A$+".0": GO T
   O R
A  IF n+p>=l THEN PRINT A$+B$( T
   O n+p-l): GO TO R
   PRINT A$( TO p+n-1)+STR$ INT
    (VAL A$(p+n TO )/10^(l-n-p)+
   .5)
R  RETURN
```

43 ARREDONDAR PARA N ALGARISMOS SIGNIFICATIVOS

```
   INPUT "Introduza o numero : "
   ; LINE A$: INPUT "Numero de"
   ;"algarismos significativos?
    ";n


   IF n=0 THEN PRINT "0": GO TO
   R
   LET B$="": LET C$="0": LET l=
   LEN A$
   FOR p=1 TO l: LET B$=B$+C$: N
   EXT P
   FOR p=1 TO l: IF A$(p)="." TH
   EN GO TO A
   NEXT p: PRINT INT (VAL A$( TO
    n+1)/10+.5);B$( TO l-n): GO
    TO R
A  IF p=1 THEN GO SUB B: PRINT I
   NT ((VAL A$)*10^(q+n-2)+.5)/
   10^(q+n-2): GO TO R
   IF n>=p-1 THEN PRINT INT ((VA
   L a$)*10^(n+1-p)+.5)/10^(n+1
   -p): GO TO R
```

```
    PRINT INT ((VAL A$)*10↑(p-n-1
     )+.5)/10↑(p-n-1): GO TO R
B   FOR q=2 TO (: IF A$(q)<>"0" T
     HEN RETURN
    NEXT q
R   RETURN
```


44 APAGAMENTO

Esta sub-rotina apaga tudo o que estiver no interior de um triângulo por nós definido.

O núcleo da rotina é muito simples e faz uso da função POINT nas linhas 17 e 22 (a contar da linha A). Esta função verifica se um *pixel* tem a cor da «tinta» (INK) ou do «fundo» (PAPER). Possui dois argumentos, colocados entre parênteses, que são as coordenadas do *pixel* em questão e o seu resultado é 0 se o *pixel* possuir a cor do fundo, e 1 no caso contrário.

Os problemas surgem por se desejar que a rotina considere todas as orientações possíveis do triângulo no plano do *écran*, incluindo aquelas que apresentam lados verticais ou horizontais. Assim, as primeiras duas linhas redefinem os vértices do triângulo, de modo a que o vértice de coordenadas com índice 3 fique a corresponder ao ponto de ordenada (y) mais elevada e o vértice de coordenadas com índice 1 ao ponto de ordenada mais baixa. O declive de cada um dos lados do triângulo é representado por m (seguido de dois índices que indicam os vértices entre os quais o lado em questão se situa), sendo utilizado o seu inverso p (também seguido de dois índices), excepto se o lado é horizontal, tomando p, neste caso, o valor zero.

O triângulo é dividido conceptualmente em duas áreas de pesquisa, indo uma do vértice inferior até à linha horizontal que passa pelo vértice de índice 2 e, a outra, desta linha até ao vértice superior. São estabelecidos dois ciclos para pesquisar cada linha de *pixels* dentro dos 2 triângulos assim formados e passar para a cor do «fundo» qualquer *pixel* que tenha a cor da «tinta», ou seja, para o qual POINT (x,y)=1. A variável t tem em conta o sentido em que cada linha é pesquisada, se da esquerda para a direita ou vice-versa e t*INT (1+ABS p31), etc., impedem que as linhas que formam os lados do triângulo sejam também apagadas. Se o sinal desta função for trocado, o próprio triângulo é eliminado.

Como qualquer forma poligonal é constituída por triângulos, esta sub-rotina pode ser adaptada e outras formas geométricas

(poligonais). De referir, contudo, a sua relativa lentidão de execução, levando o triângulo da figura de exemplo cerca de 3 minutos a ter o seu «interior» completamente apagado. O parágrafo (A$) impresso repetidamente como fundo serve apenas para visualizar o processo de apagamento.

```
      LET A$="   Esta e' uma rotina
       de apaga- mento que apaga o
       s pixels no in-terior de um
       triangulo usando asfuncoes P
       OINT e PLOT OVER 1"
      FOR p=2 TO 20 STEP 4: PRINT O
       VER 1;AT p,0;A$: NEXT p
      INPUT "Vertices do triangulo:
       "''"x1=";x1,"y1=";y1''"x2=";x
       2,"y2=";y2''"x3=";x3,"y3=";y3
      PLOT x1,y1: DRAW x2-x1,y2-y1:
       DRAW x3-x2,y3-y2: DRAW x1-x3
       ,y1-y3

A     IF y2>y3 THEN LET a=y3: LET b
       =x3: LET y3=y2: LET x3=x2: L
       ET y2=a: LET x2=b
      IF y1>y2 THEN LET a=y2: LET b
      =x2: LET y2=y1: LET x2=x1: L
       ET y1=a: LET x1=b: GO TO A
      IF x1<>x2 THEN LET m12=(y1-y2
       )/(x1-x2): GO TO B
      LET p12=0: GO TO C
B     IF y1<>y2 THEN LET p12=1/m12
C     IF x2<>x3 THEN LET m23=(y2-y3
       )/(x2-x3): GO TO D
      LET p23=0: GO TO E
D     IF y2<>y3 THEN LET p23=1/m23
E     IF x3<>x1 THEN LET m31=(y3-y1
       )/(x3-x1): GO TO F
      LET p31=0: GO TO G
F     IF y3<>y1 THEN LET p31=1/m31
G     IF y1=y2 THEN GO TO H
      LET t=SGN (p12-p31)
      FOR y=y1 TO y2-(y3=y2)
      FOR x=x1-y1*p31+y*p31+t*INT (
       1+ABS p31) TO x1-y1*p12+y*p1
       2-t*INT (1+ABS p12) STEP t
      IF POINT (x,y)=1 THEN PLOT OV
       ER 1;x,y
      NEXT x: NEXT y
H     IF y2=y3 THEN GO TO R
      LET t=SGN (p31-p23)
```



```
      FOR y=y2+(y1=y2) TO y3
      FOR x=x1-y1*p31+y*p31+t*INT (
       1+ABS p31) TO x2-y2*p23+y*p2
       3-t*INT (1+ABS p23) STEP t
      IF POINT (x,y)=1 THEN PLOT OV
       ER 1;x,y
      NEXT x: NEXT y
R     RETURN
```

Exemplo:

45 POUPANÇA DE MEMÓRIA

Elaborar programas de forma a que estes ocupem um espaço mínimo de memória é um objectivo que se vai tornando cada vez mais importante à medida que os nossos conhecimentos de programação aumentam e os programas se vão tornando maiores e mais complexos. No *Spectrum* de 16 k é muito fácil escrever um programa que ocupe a totalidade da memória, mas é também relativamente simples poupar uma área de memória muito considerável por meio de uma programação cuidada. Apresentam-se a seguir algumas das formas pelas quais é possível fazer isto.

Além do processo que consiste na utilização de CLEAR para deslocar a RAMTOP para cima de modo a obter mais espaço de memória disponível, uma das formas mais simples de conseguir o objectivo proposto é evitar a utilização de números, os quais necessitam de cinco *bytes* de memória para serem armazenados.

Se incluírmos num programa a instrução PRINT (PEEK 23653+256*PEEK 23654) para indicar o início da área livre de memória, podemos ver como a memória vai sendo preenchida.

Para evitarmos usar números directamente, podemos empregar, por exemplo,

```
LET x=VAL "5" em vez de LET x=5
LET y=CODE "d" em vez de LET y=100
LET z=INT PI      em vez de LET z=3
```

Podemos gerar números pequenos da forma seguinte:

		«Bytes» poupados
−1	LET a=COS PI	6
0	LET b=SIN PI	4
1	LET c=COS SIN PI	4
2	LET d=VAL "2" ou d=c+c	3 ou 4
3	LET e =INT PI	5

Quando um número é usado muitas vezes, como em ciclos, que frequentemente começam em 1, podemos utilizar o seguinte:

```
LET j=COS SIN PI          e, depois,
FOR p=j TO VAL "9"        por exemplo.
```

Devemos também usar a mesma variável para ciclos não «encaixados» *(non nested loops)*, a não ser que queiramos utilizar o valor final da variável em quaisquer outras operações.

Assim, por exemplo,

```
   10 FOR a=1 TO 10: PRINT a: NEX
T a
   20 FOR b=1 TO 20: PRINT b: NEX
T b

ocupa mais 19 bytes do que

   10 FOR a=1 TO 10: PRINT a: NEX
T a
   20 FOR a=1 TO 20: PRINT a: NEX
T a
```

De um modo geral, devemos utilizar o menor número possível de nomes de variáveis, pois o último valor que a variável assume encontra-se sempre em memória.

Podemos atribuir valores a variáveis sem utilizarmos números de linhas para as respectivas declarações LET, ou seja, sem incluir estas declarações no corpo do programa. Assim, elas ficam com os respectivos valores em memória, mas sem utilizar o espaço extra ocupado pelas declarações LET e pelos números de linha.

Assim, por exemplo, em vez de

```
10 LET x=5 : LET y=10 etc.
```

digitar

```
LET x=5 : LET y=10 etc. e ENTER
```


Contudo, como RUN apaga os valores das variáveis, devemos, neste caso, utilizar GO TO 1 para fazer executar o programa.

Devemos, igualmente, definir quadros o mais longe possível do início do programa e utilizar nomes de quadros já utilizados anteriormente, se estes já não forem necessários. Por exemplo, se DIM A(m,n) for utilizado na parte inicial de um programa para definir um ou mais quadros que a partir de certo ponto não sejam mais necessários, poderemos utilizar DIM A(p,q) para qualquer outro quadro que o seja. Devemos posteriormente fazer DIM A(1) se o quadro A não vai ser utilizado de novo. Lembremo-nos de que os quadros ocupam muito espaço de memória. Por exemplo, DIM A(3,20) vai definir um quadro A, dimensionando-o para 3 colunas×20 linhas, quadro esse que vai ocupar 318 *bytes*.

O espaço de memória ocupado por quadros de cadeias alfanuméricas *(string arrays)* é também bastante importante, embora comparativamente menor do que o exigido por quadros numéricos. Assim, por exemplo, DIM A$(3,20) só ocupa cerca de 70 *bytes*.

No mesmo intuito de poupança de memória, devemos empregar o menor número possível de números de linha, utilizando declarações múltiplas separadas por dois pontos (separador de declarações). Contudo, temos de observar as seguintes excepções:

Não utilizar declarações múltiplas:

a) depois de uma declaração IF (excepto se o desejarmos, pois ficariam dependentes dessa declaração);
b) depois de uma declaração REM (não seriam executadas). As declarações REM devem ser evitadas, pois ocupam inutilmente espaço de memória, já que não são executadas;
c) onde for necessário o acesso a um comando NEXT num ciclo encaixado.

Devemos também utilizar, sempre que possível, sub-rotinas nos programas, para evitar a repetição de uma mesma operação ou conjunto de instruções.

Em relação a este ponto, é conveniente utilizarmos contadores

de ciclo em operações GOSUB que empreguem números de linha (os quais correspondem ao início de cada sub-rotina) incluídos nas próprias declarações GOSUB. Assim, por exemplo, em vez do programa

```
1Ø LET t=Ø
2Ø Operação qualquer : LET t=t+1
3Ø IF t=1 THEN GOSUB 1ØØØ
4Ø IF t=2 THEN GOSUB 1Ø5Ø
5Ø IF t=3 THEN GOSUB 11ØØ
6Ø IF t<4 THEN GO TO 2Ø
```

podemos, com vantagem evidente, usar o seguinte:

```
1Ø LET t=95Ø
2Ø Operação qualquer : LET t=t+5Ø
3Ø IF t<115Ø THEN GOSUB t : GO TO 2Ø
```

Para as 223 primeiras linhas do programa, podemos substituir os números de linha (desde que inferiores a 223, claro) incluídos em declarações GOSUB e GO TO pelo código ASCII (32 a 255) de um carácter ou palavra de comando *(keyword)* entre SPACE (32) e COPY (255), do modo seguinte:

```
GOSUB CODE «carácter ASCII 32 a 255»
ou
GO TO CODE «carácter ASCII 32 a 255»
```

A instrução CODE vai dar o número de código ASCII do carácter escrito entre parênteses, sendo CODE «carácter ASCII» substituído por esse número. Assim, por exemplo, podemos substituir

```
GOSUB 1ØØ por GOSUB CODE "d"
ou
GOSUB 25Ø por GOSUB CODE "IF"
```

De notar que «IF», no segundo exemplo, é uma palavra de

comando e deve ser introduzida como tal. Para o fazer, introduzir primeiramente GOSUB CODE : IF e depois apagar os dois pontos e acrescentar as aspas. O mesmo se pode dizer para quaisquer outras palavras de comando incluídas no conjunto de caracteres do *Spectrum* (embora estas não façam propriamente parte do código ASCII, assim como os caracteres gráficos e outros tipos de caracteres próprios do *Spectrum*).

O conjunto de caracteres é apresentado, juntamente com o código de cada carácter, no apêndice A do manual de programação BASIC do *Spectrum*, mas pode ser visualizado no *écran* (códigos 32 a 255) se fizermos executar o seguinte programa:

```
1Ø FOR a=32 TO 255 : PRINT CHR$ a; " ";: NEXT a
```

A vantagem da substituição do número de linha pelo código de carácter equivalente reside na já mencionada poupança de *bytes* quando não se utilizam números directamente.

46 SISTEMAS DE EQUAÇÕES

Esta sub-rotina calcula a solução de um sistema com um máximo de oito equações lineares a oito incógnitas. O método utilizado não está, contudo, limitado a oito equações, sendo esta limitação decorrente apenas da sub-rotina de validação de *input* (14), com a qual a presente sub-rotina está feita para funcionar. Portanto, não é difícil superar tal limitação, se assim desejarmos.

O método empregado nesta sub-rotina baseia-se no cálculo matricial. Assim, por exemplo, um sistema de 3 equações lineares a 3 incógnitas,

```
a1*x+b1*y+c1*z=k1
a2*x+b2*y+c2*z=k2
a3*x+b3*y+c3*z=k3
```

pode ser representado em notação matricial da seguinte forma (segundo Cauchy):

$$\begin{bmatrix} a1 & b1 & c1 \\ a2 & b2 & c2 \\ a3 & b3 & c3 \end{bmatrix} * \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} k1 \\ k2 \\ k3 \end{bmatrix}$$

donde

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} a1 & b1 & c1 \\ a2 & b2 & c2 \\ a3 & b3 & c3 \end{bmatrix}^{-1} * \begin{bmatrix} k1 \\ k2 \\ k3 \end{bmatrix}$$

Designando por E a matriz dos coeficientes numéricos, E^{-1} será a sua matriz inversa, a qual é calculada pelo método de Gauss-Jordan (ver sub-rotinas 21 e 28) e multiplicada depois pela matriz dos termos independentes (ki) para dar a solução do sistema, a matriz-coluna x,y,z.

A segunda linha da sub-rotina propriamente dita verifica se o coeficiente a1 é zero e permuta duas linhas em caso afirmativo, por meio da sub-rotina A. Isto destina-se a evitar que a sub-rotina «estoire» na segunda declaração da linha C, ao ser tentada uma divisão por zero. A sub-rotina de inversão da matriz dos coeficientes inicia-se na citada linha C e destina-se também a verificar se o determinante da matriz em questão é diferente de zero (condição necessária para que a matriz seja invertível). Após ter sido executada esta sub-rotina, a matriz inversa, por ela acabada de calcular, é multiplicada pela matriz dos termos independentes, E(k,n+1) e o resultado, os elementos S(j) da matriz-coluna S, é impresso no *écran*, utilizando CHR$ (90−n+j) para imprimir os nomes adequados das variáveis (x, y, z, etc.).

A sub-rotina verifica se o sistema tem solução e, caso afirmativo, resolve-o.

```
INPUT "Numero de variaveis? "
 ;n: DIM E(n,n+1)
FOR j=1 TO n: FOR k=1 TO n+1
INPUT "Introduza os coeficien
 tes:";"E(";j;",";k;")=";
 E(j,k)
NEXT k: NEXT j
```

```
Se n<8, utilizar como alterna-
tiva a sub-rotina 14 (Input de da-
dos para equacoes lineares com
um maximo de 8 variaveis) para
introduzir os valores de n e de
E(n,n+1), como no exemplo.
```

```
LET d=1
IF E(1,1)=0 THEN GO SUB A
GO SUB C: DIM S(n): FOR j=1 T
 O n: LET S(j)=0
FOR k=1 TO n: LET S(j)=S(j)+E
 (j,k)*E(k,n+1)
NEXT k: NEXT j: GO TO R
A  FOR j=2 TO n: IF E(j,1)<>0 TH
    EN LET k=j: GO TO B
   NEXT j: GO TO F
```



```
B  FOR j=1 TO n+1: LET z=E(1,j):
    LET E(1,j)=E(k,j): LET E(k,j
   )=z: NEXT j: RETURN
C  FOR i=1 TO n: LET E(i,i)=1/E(
   i,i)
   FOR j=1 TO n: IF j=i THEN GO
    TO D
   LET E(j,i)=E(j,i)*E(i,i)
   FOR k=1 TO n: IF k=i THEN GO
    TO E
   LET E(j,k)=E(j,k)-E(j,i)*E(i,
   k)
   IF j<>n-(i=n) THEN GO TO E
   LET E(i,k)=-E(i,i)*E(i,k)
E  NEXT k
D  NEXT j: IF i<>n THEN IF ABS E
    (i+1,i+1)<=1E-8 THEN GO TO F
   NEXT i: RETURN
F  LET d=0
R  RETURN
```

```
R  IF d=0 THEN PRINT "O sistema
    nao tem solucao": STOP
   FOR p=7 TO 19: PRINT AT 1,7;"
    A solucao e':"
   POKE 16448+p,255: NEXT p: PRI
    NT
   FOR j=1 TO n
   PRINT TAB 3;CHR$ (90-n+j);"="
    ;S(j): PRINT
   NEXT j
```

Exemplo

```
Qual o numero de variaveis? 6
```

As equacoes sao :

U/Y	U/Z	W	X	Const
a1*U +	b1*U +	c1*W +	d1*X +	
e1*Y +	f1*Z			= K1
a2*U +	b2*U +	c2*W +	d2*X +	
e2*Y +	f2*Z			= K2
a3*U +	b3*U +	c3*W +	d3*X +	
e3*Y +	f3*Z			= K3

```
a4*U + b4*V + c4*W + d4*X + 
e4*Y + f4*Z                = K4
a5*U + b5*V + c5*W + d5*X + 
e5*Y + f5*Z                = K5
a6*U + b6*V + c6*W + d6*X + 
e6*Y + f6*Z                = K6
```

```
Introduza os valores numericos:
a1=?
```

As equacoes sao :

U/Y	U/Z	W	X	Const
7 +	5 +	-4 +	2 +	
13 +	6			= 15
4.5 +	2 +	19 +	24 +	
-7 +	9.6			= 3
5.5 +	13 +	9.43 +	-5.46 +	
5 +	32			= 5.44
-7.9 +	6.1 +	3.2 +	2.7 +	
5 +	8			= 23
6.9 +	5.3 +	6.7 +	2 +	
0 +	5.7			= 4.8
-5.4 +	1 +	6.8 +	4 +	
3.88 +	4			= 87

A solucao e':

U=-1.4395573

V=-7.1904258

W=10.764237

X=-3.9475245

Y=9.5356364

Z=-1.9970411

Nota: O tempo de cálculo para este exemplo é de cerca de 1Ø segundos.

Seguidamente, apresenta-se a listagem completa da conjugação das sub-rotinas 14 e 46, tal como foi utilizada para o cálculo do exemplo anterior. De notar a inclusão da linha «extra» 16Ø, imprescindível à boa apresentação gráfica dos resultados.

```
   5 REM "VALIDACAO DE INPUT" -
    Sistemas de Equacoes Lineares
  10 INPUT "Qual o numero de var
iaveis? ";n
  15 IF n<1 OR n>8 THEN GO TO 10
  20 PRINT AT 0,6;"A equacao e':
" AND (n=1);"As equacoes sao : "
 AND (n>1)
  30 FOR p=6 TO 18+(n<>1)*4: POK
E 16416+p,255: NEXT p: DIM E(n,n
+1)
  40 FOR j=1 TO n: FOR k=1 TO n
  50 PRINT AT 2*j+2+(k)4),7*k-7-
28*(k>4);CHR$ (96+k)+STR$ j+"*"+
CHR$ (90+*k-n)
  60 NEXT k: FOR k=1 TO n-1
  70 PRINT AT 2*j+2+(k>4),7*k-2-
28*(k>4);"+": NEXT k
  80 PRINT AT 2*j+2+(k>4),25;" =
 "+"K"+STR$ j: NEXT j
  90 FOR k=1 TO n: PRINT AT 2,27
;"Const";AT 2,7*k-7-26*(k>4);CHR
$ (32+15*(k>4))+CHR$ (90+k-n): N
EXT k
 100 FOR j=1 TO n: FOR k=1 TO n+
1
 110 INPUT "Introduza os valores
 numericos ";(CHR$ (96+k-(k=n+1)
*(+21))+STR$ j);"=? ";E(j,k)
 120 PRINT AT 2*j+2+(n)4)*(k>4),
(7*k-7-28*(k>4))*(k<>n+1)+(k=n+1
)*28;"      ";CHR$ 8;CHR$ 8;CHR$ 8
;CHR$ 8;E(j,k)
 130 NEXT k: NEXT j
 140 INPUT "          OK? (s/n) "
;k$
 150 IF k$<>"s" THEN GO TO 100
 160 CLS
 190 REM
```

Sub-rotina 46
Sistema de Equacoes

```
 200 LET d=1
 210 IF E(1,1)=0 THEN GO SUB 250
 220 GO SUB 280: DIM S(n): FOR j
=1 TO n: LET S(j)=0
 230 FOR k=1 TO n: LET S(j)=S(j)
+E(j,k)*E(k,n+1)
 240 NEXT k: NEXT j: GO TO 400
 250 FOR j=2 TO n: IF E(j,1)<>0
THEN LET k=j: GO TO 270
 260 NEXT j: GO TO 390
 270 FOR j=1 TO n+1: LET z=E(1,j
): LET E(1,j)=E(k,j): LET E(k,j)
=z: NEXT j: RETURN
 280 FOR i=1 TO n: LET E(i,i)=1/
E(i,i)
 290 FOR j=1 TO n: IF j=i THEN G
O TO 360
 300 LET E(j,i)=E(j,i)*E(i,i)
 310 FOR k=1 TO n: IF k=i THEN G
O TO 350
 320 LET E(j,k)=E(j,k)-E(j,i)*E(
i,k)
 330 IF j<>n-(i=n) THEN GO TO 35
0
 340 LET E(i,k)=-E(i,i)*E(i,k)
 350 NEXT k
 360 NEXT j: IF i<>n THEN IF ABS
E(i+1,i+1)<=1E-8 THEN GO TO 380
 370 NEXT i: RETURN
 380 LET d=0
 400 IF d=0 THEN PRINT "O sistem
a nao tem solucao": STOP
 410 FOR p=7 TO 19: PRINT AT 1,7
;"A solucao e':"
 420 POKE 16448+p,255: NEXT p: P
RINT
 430 FOR j=1 TO n
 440 PRINT TAB 3;CHR$ (90-n+j);"
=";S(j): PRINT
 450 NEXT j
```


47-50 SÉRIES

Uma série é a sucessão formada pelas somas sucessivas dos termos de uma dada sucessão. Assim, seja u1, u2, u3, . . . uma sucessão dada; a sucessão S1, S2, S3, . . . onde

$$S1=u1,\quad S2=u1+u2,\quad S3=u1+u2+u3,\ldots,\quad Sn=u1+u2+u3+\ldots+un,\ldots$$

será a série formada a partir da sucessão dada, e Sn é a soma dos n primeiros termos dessa sucessão, designando-se essa soma por «termo geral» ou «termo de ordem n» da série. Uma série diz-se «finita» quando tem um número limitado de termos e «infinita» no caso contrário. Além disso, se o $\lim_{n\to\infty} Sn=S$ (limite, quando n tende para infinito, de Sn) existe, a série diz-se «convergente», sendo S a soma da série. Caso contrário, a série diz-se «divergente».

Este grupo de sub-rotinas calcula a soma dos n primeiros termos (Sn) para várias séries importantes, nomeadamente as séries exponencial, geométrica, aritmética e binomial.

Ora as séries podem, geralmente, ser reescritas na forma de uma operação de ciclo, ou cíclica, a qual é mais rápida de calcular do que um conjunto de termos individuais. Assim, a série exponencial, dada pela sua soma

$$S=1+x+x^2/2!+x^3/3!+\ldots$$

pode ser reescrita como

$$S=1+\frac{x}{2}(1+\frac{x}{3}(1+\frac{x}{4}(1+\ldots$$

a série geométrica, dada por

$$S=a+a*r+a*r^2+a*r^3+\ldots$$

como

$$S=a(1+r(1+r(1+r(1+\ldots$$

a série aritmética, dada por

$$S=a+(a+d)+(a+2*d)+(a+3*d)+\ldots$$

como

$$S=n*a+d(1+2+3+4+\ldots$$

e a série binomial, dada por

$$S=1+\frac{nx}{1!}+\frac{n*(n-1)*x^2}{2!}+\frac{n*(n-1)*(n-2)*x^3}{3!}+\ldots$$

como

$$S=1+\frac{nx}{1}\left(1+\frac{(n-1)}{2}\times\left(1+\frac{(n-2)}{3}\times(1+\ldots\right.\right.$$

A forma de cada ciclo torna-se óbvia se analisarmos a respectiva expressão «normal» de S.

47 SÉRIE EXPONENCIAL

$$S=1+x+x^2/2!+x^3/3!+\ldots=e^x$$

```
    INPUT n,x

    LET f=1: LET e=1
    FOR p=1 TO n: LET f=f*X/p: LE
     T e=e+f
    NEXT p
R   RETURN

R   PRINT TAB 8;"EXP ";x;"=";e
```

48 SÉRIE GEOMÉTRICA

$$S=a+a*r+a*r^2+a*r^3+\ldots=\frac{a}{1-r}\ \text{se}\ |r|<1$$

$$Sn=\frac{a(1-r^n)}{1-r}$$

```
    INPUT a: INPUT r: INPUT n

    LET S=0: FOR p=1 TO n: LET S=
     S+a*r↑(p-1): NEXT p
R   RETURN

R   PRINT "S=";S
```

49 SÉRIE ARITMÉTICA

$$S=a+(a+d)+(a+2*d)+(a+3*d)+\ldots$$

```
    INPUT a: INPUT d: INPUT n

    LET S=0: FOR p=1 TO n: LET S=
     S+a+(p-1)*d: NEXT p
R   RETURN

R   PRINT "S=";S
```

50 SÉRIE BINOMIAL

$$S=1+\frac{n*x}{1!}+\frac{n*(n-1)*x^2}{2!}+\frac{n*(n-1)*(n-2)*x^3}{3!}+\ldots$$

$Sn=(1+x)^n$ se n for inteiro positivo.

```
    INPUT "x=";x;" n=";n;"  No. d
     e termos r=";r

    LET S=1: LET Z=1: LET m=n
    FOR p=1 TO r: LET Z=Z*m*x/p
    LET S=S+Z: LET m=m-1
    NEXT p
R   RETURN

R   PRINT "S=";S
```


51 IMPRESSÃO TABELADA

Esta sub-rotina permite imprimir números em posições de impressão predefinidas (PRINT AT), mas de modo a que, caso o número seja bastante grande e/ou seja impresso junto à margem direita do *écran*, os seus últimos algarismos sejam escritos por baixo dos primeiros e não no extremo oposto do *écran* (início da linha seguinte). A este respeito, experimente-se a sub-rotina (com números de linha, evidentemente) para n=2504562377, L=5 e c=24, por exemplo.

A sub-rotina faz uso da variável de sistema 23688 (S POSN), a qual contém 33 menos o número da coluna da posição de impressão corrente, no *écran*. A execução baseia-se em dois ciclos, comandados respectivamente por FOR p=1 TO LEN n$ e por LET c=c+1. Quando a posição de impressão corrente atinge a margem direita do *écran*, c passa a 32+p−LEN n$ para que o resto do número (n$) possa ser impresso.

```
    INPUT "Numero? ";n,"Linha? ";
     l,"Coluna? ";c

    LET n$=STR$ n
    FOR p=1 TO LEN n$: PRINT AT l
     ,c;n$(p);
    LET c=c+1
    IF PEEK 23688=1 THEN LET l=l+
     1: LET c=32+p-LEN n$
    NEXT p
R   RETURN
```

52 ANÁLISE ESTATÍSTICA

Esta sub-rotina calcula várias «propriedades» ou medidas estatísticas para um grupo de dados, nomeadamente a média, o desvio-padrão, o mínimo e máximo e o número de dados incluídos em cada intervalo de classe do histograma, e apresenta os resultados sob a forma de um histograma «tridimensional» ou 3-D.

O número de intervalos de classe no histograma é calculado de forma a ficar próximo do valor da raiz quadrada do número de dados, para que a apresentação gráfica dos resultados seja satisfatória.

As linhas 1 a 8 da sub-rotina propriamente dita calculam o máximo, o mínimo, a média e o desvio-padrão (S) do conjunto de dados (ver, a este respeito, a sub-rotina 30). Estes valores são, seguidamente, armazenados em Z$, para que se possa controlar o número de dígitos impressos no *écran* (por segmentação da cadeia) no final do programa.

O quadro D() é utilizado para armazenar:

a) o valor mais baixo dos intervalos de classe do histograma, em D(1,i+3);
b) o número de valores de cada intervalo, em D(2,i+3).

O maior dos números de valores é então atribuído à variável D2max.

Depois de o *écran* ter sido «limpo» (quinta linha depois da linha A), é impresso o cabeçalho e, seguidamente, sublinhado. O quadro A(), definido e dimensionado por DIM A(2,i+4), armazena os valores do histograma a desenhar, os quais são afectados de um factor de escala de modo a ajustarem-se a uma linha com um declive de 1/3 que passa pelos pontos de coordenadas (92,36) e (188,68), pontos esses que vão representar, respectivamente, os valores mínimo e máximo, no gráfico do histograma. A cota vertical (ordenada) é afectada de um factor de escala de 50 pela

linha 34 (terceira depois de F). As linhas seguintes destinam-se a desenhar o histograma e a sombreá-lo. As linhas a seguir a B desenham e graduam os eixos principais.

```
      INPUT "Numero de dados (min 4
       )? ";n: DIM V(n)
      FOR p=1 TO n
      INPUT "V";(p);"=";V(p)
      NEXT p


Utilizar, como alternativa a sub
rotina 16 de validacao de input
(uma variavel) para introduzir
os valores de n e de V(n).



      LET i=INT SQR n: LET xm=0: LE
       T S=0
      LET xmax=0: LET xmin=V(1)
      FOR p=1 TO n
      LET xm=xm+V(p): IF V(p)>=xmax
       THEN LET xmax=V(p)
      NEXT p: LET xm=xm/n
      FOR p=1 TO n
      LET S=S+(V(p)-xm)*(V(p)-xm):
       IF V(p)>xmax THEN LET xmax=V
       (p)
      NEXT p: LET S=SQR (S/(n-1))
      LET t=(xmax-xmin)/i: LET xf=t
       *i
      DIM Z$(i+9,9): FOR p=5 TO i+7
      LET Z$(p, TO 9)=STR$ (xmin+(p
       -6.5)*t): NEXT p
      LET Z$(1, TO 9)=STR$ xm
      LET Z$(2, TO 9)=STR$ xmin
      LET Z$(3, TO 9)=STR$ xmax
      LET Z$(4, TO 9)=STR$ S
      LET Z$(5, TO 9)=""
      DIM D(2,i+3): LET D2max=0: LE
       T D(2,1)=0
      FOR p=1 TO n: FOR q=5 TO i+7
      IF V(p)<=VAL Z$(q, TO 4) THEN
       LET D(2,q-5)=D(2,q-5)+1: GO
       TO A
      LET D(2,i+3)=0: NEXT q
A     NEXT p
      FOR p=1 TO i+2
      LET D(1,p)=VAL Z$(p+5, TO 4):
       IF D(2,p)>=D2max THEN LET D2
```

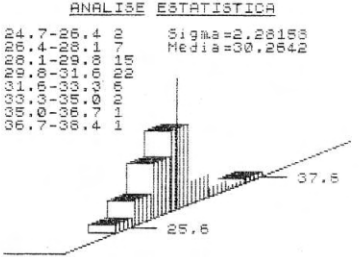


```
       max=D(2,p)
      NEXT p
      CLS : PRINT AT 0,6;"ANALISE E
       STATISTICA"
      FOR p=6 TO 24: IF p<>13 THEN
       POKE 16416+p,255
      NEXT p: PRINT OVER 1
      DIM A(2,i+4): FOR p=2 TO i+4
      LET A(1,p)=((D(1,p-1)-xmin)*1
       01.1928/xf+48)*COS 0.32175
      LET A(2,p)=A(1,p)*TAN 0.32175
       : NEXT p
      FOR p=2 TO i+3: LET q=0
F     IF A(1,p)+q>A(1,p+1) THEN GO
       TO B
      IF p<=i+2 THEN PLOT 44+A(1,p)
       +q,20+A(2,p)+q*TAN 0.32175
      DRAW 0,50/D2max*D(2,p)
      IF D(2,p)>=D(2,p-1) THEN DRAW
       -20,0: GO TO C
      IF p<i+3 AND q<20 THEN DRAW -
       q,0: GO TO D
      IF p<i+3 AND q>20 THEN DRAW -
       20,0
D     IF D(2,p)<=D(2,p-1) THEN GO T
       O E
C     IF q=0 THEN DRAW 0,-50/D2max*
       (D(2,p)-D(2,p-1))
E     LET q=q+3: GO TO F
B     NEXT p
      PLOT 44,20: DRAW 211,211*TAN
       0.32175
      PLOT 44,20: DRAW -44,0
      PLOT 44+A(1,2),20+A(2,2): DRA
       W -20,0
      FOR p=1 TO i+1: PRINT AT 1+p,
       0;Z$(p+5, TO 4);"-";Z$(p+6,
       TO 4);" ";D(2,p+1): NEXT p
      PRINT AT 2,15;"Sigma=";Z$(4,
       TO 7)
      PRINT AT 3,15;"Media=";Z$(1,
       TO 7)
      PRINT AT 17,15;Z$(2, TO 4)
      PRINT AT 18,27;Z$(3, TO 5)
      PLOT 92,36: DRAW 20,0
      PLOT 188,56: DRAW 20,0
      PLOT 44+((xm-xmin)*101.193/xf
       +48)*COS 0.32175,20+((xm-xmi
       n)*101.193/xf+48)*SIN 0.3217
       5: DRAW 0,80
R     RETURN
```

Exemplo 1

Os dados da tabela seguinte produzem o histograma da figura apresentada abaixo:

V1 = 25.6	V15 = 35.0	V29 = 27.9	V43 = 28.9
V2 = 28.3	V16 = 27.1	V30 = 32.1	V44 = 32.0
V3 = 30.1	V17 = 29.3	V31 = 28.7	V45 = 27.9
V4 = 26.9	V18 = 30.6	V32 = 29.1	V46 = 31.0
V5 = 37.6	V19 = 30.5	V33 = 32.1	V47 = 30.5
V6 = 30.8	V20 = 29.5	V34 = 31.3	V48 = 29.9
V7 = 26.0	V21 = 28.0	V35 = 30.0	V49 = 30.1
V8 = 29.3	V22 = 33.1	V36 = 29.6	V50 = 30.3
V9 = 30.2	V23 = 36.0	V37 = 28.7	V51 = 28.7
V10 = 31.6	V24 = 29.8	V38 = 35.0	V52 = 31.2
V11 = 28.7	V25 = 31.2	V39 = 31.2	V53 = 33.1
V12 = 29.5	V26 = 30.1	V40 = 27.9	V54 = 28.9
V13 = 30.3	V27 = 28.7	V41 = 30.5	V55 = 27.9
V14 = 32.4	V28 = 31.5	V42 = 31.6	V56 = 31.0



Exemplo 2

Associação das rotinas 16 e 52

```
    1 REM 52 ANALISE ESTATISTICA

    5 REM "VALIDACAO DE INPUT"
              - Uma variavel -
  10 INPUT "Qual o numero de dad
os? ";n
  20 IF n>=4 THEN GO TO 50
  30 PRINT n;" dados e'um numero
 insuficientepara efectuar uma a
nalise esta- tistica correcta,"
""Pressione qualquer tecla para
 recomecar.": GO SUB 220: CLS
  40 GO TO 10
  50 DIM V(n)
  60 FOR p=1 TO n
  70 INPUT "Introduza os dados:
V";(p);"="; LINE V$: GO SUB 250
  80 IF t=LEN V$+1 THEN LET V(p)
=VAL V$: GO TO 100
  90 GO SUB 270: GO TO 70
 100 NEXT p
 110 PRINT "Estes dados estao co
rrectos? "
 120 GO SUB 130: GO TO 140
 130 PRINT : FOR p=1 TO n: PRINT
 "V";p;"=";V(p): NEXT p: RETURN
 140 INPUT "(s/n)? ";k$
 150 IF k$="s" THEN GO TO 280
 160 INPUT "Que dado quer altera
r-Linha? "; LINE V$: GO SUB 250:
 IF t=LEN V$+1 THEN LET p=VAL V$
: GO TO 180
 170 GO SUB 270: GO TO 150
 180 IF p<=n THEN GO TO 200
 190 PRINT "Esse numero nao faz
a parte dos dados originais. Pri
ma qualquer tecla para voltar at
ras.": GO SUB 220: GO TO 160
 200 INPUT "Introduza o valor co
rrecto para V";(p);"=? "; LINE V
$: GO SUB 250: IF t=LEN V$+1 THE
N LET V(p)=VAL V$: CLS : GO TO 1
10
 210 GO SUB 270: GO TO 200
 220 IF INKEY$<>"" THEN GO TO 22
0
 230 IF INKEY$="" THEN GO TO 230
 240 RETURN
 250 FOR t=1 TO LEN V$: IF V$(t)
```

```
)="0" AND V$(t)<="9" OR V$(t)=
" OR V$(t)="+" OR V$(t)="-" THEN
 NEXT t
 260 CLS : RETURN
 270 PRINT V$;" nao e'um numero"
: RETURN
 280 REM


          ANALISE ESTATISTICA

 290 CLS : PRINT AT 10,3; FLASH
1;"CALCULOS A SEREM EFECTUADOS"
 300 LET i=INT 50R n: LET Xm=0:
LET S=0
 310 LET xmax=0: LET xmin=V(1)
 320 FOR p=1 TO n
 330 LET xm=xm+V(p): IF V(p)>=xm
ax THEN LET xmax=V(p)
 340 NEXT p: LET xm=xm/n
 350 FOR p=1 TO n
 360 LET S=S+(V(p)-xm)*(V(p)-xm)
: IF V(p)>xmax THEN LET xmax=V(p
)
 370 NEXT p: LET S=50R (S/(n-1))
 380 LET t=(xmax-xmin)/i: LET xr
=t*i
 390 DIM Z$(i+9,9): FOR p=6 TO i
+7
 400 LET Z$(p, TO 9)=STR$ (xmin+
(p-6.5)*t): NEXT p
 410 LET Z$(1, TO 9)=STR$ xm
 420 LET Z$(2, TO 9)=STR$ xmin
 430 LET Z$(3, TO 9)=STR$ xmax
 440 LET Z$(4, TO 9)=STR$ S
 450 LET Z$(5, TO 9)=""
 460 DIM D(2,i+3): LET D2max=0:
LET D(2,1)=0
 470 FOR p=1 TO n: FOR q=6 TO i+
7
 480 IF V(p)<=VAL Z$(q, TO 4) TH
EN LET D(2,q-5)=D(2,q-5)+1: GO T
O 500
 490 LET D(2,i+3)=0: NEXT q
 500 NEXT p
 510 FOR p=1 TO i+2
 520 LET D(1,p)=VAL Z$(p+5, TO 4
): IF D(2,p)>D2max THEN LET D2m
ax=D(2,p)
 530 NEXT p
 540 CLS : PRINT AT 0,6;"ANALISE
```



```
 ESTATISTICA"
 550 FOR p=6 TO 24: IF p<>13 THE
N POKE 16416+p,255
 560 NEXT p: PRINT OVER 1
 570 DIM A(2,i+4): FOR p=2 TO i+
4
 580 LET A(1,p)=((D(1,p-1)-xmin)
*101.1928/xr+48)*COS 0.32175
 590 LET A(2,p)=A(1,p)*TAN 0.321
75: NEXT p
 600 FOR p=2 TO i+3: LET q=0
 610 IF A(1,p)+q>A(1,p+1) THEN G
O TO 700
 620 IF p<=i+2 THEN PLOT 44+A(1,
p)+q,20+A(2,p)+q*TAN 0.32175
 630 DRAW 0,50/D2max*D(2,p)
 640 IF D(2,p)>=D(2,p-1) THEN DR
AW -20,0: GO TO 680
 650 IF p<i+3 AND q<20 THEN DRAW
-q,0: GO TO 670
 660 IF p<i+3 AND q>20 THEN DRAW
-20,0
 670 IF D(2,p)<=D(2,p-1) THEN GO
TO 690
 680 IF q=0 THEN DRAW 0,-50/D2ma
x*(D(2,p)-D(2,p-1))
 690 LET q=q+3: GO TO 610
 700 NEXT p
 710 PLOT 44,20: DRAW 211,211*TA
N 0.32175
 720 PLOT 44,20: DRAW -44,0
 730 PLOT 44+A(1,2),20+A(2,2): D
RAW -20,0
 740 FOR p=1 TO i+1: PRINT AT 1+
p,0;Z$(p+5, TO 4);"-";Z$(p+6, TO
 4);" ";D(2,p+1): NEXT p
 750 PRINT AT 2,15;"Sigma=";Z$(4
, TO 7)
 760 PRINT AT 3,15;"Media=";Z$(1
, TO 7)
 770 PRINT AT 17,15;Z$(2, TO 4)
 780 PRINT AT 18,27;Z$(3, TO 5)
 790 PLOT 92,36: DRAW 20,0
 800 PLOT 188,68: DRAW 20,0
 810 PLOT 44+((xm-xmin)*101.193/
xr+48)*COS 0.32175,20+((xm-xmin)
*101.193/xr+48)*SIN 0.32175: DRA
W 0,80
 820 STOP
```

53 TESTE PARA UM NÚMERO DECIMAL

Esta sub-rotina testa um número decimal positivo ou negativo, com ou sem ponto decimal (vírgula).

Como o número é introduzido sob a forma de uma cadeia, cada carácter (ou algarismo, neste caso) pode ser testado separadamente. Se for utilizado o sinal + ou o sinal −, estes têm de ser introduzidos em primeiro lugar, como primeiro carácter do número, e têm de ser testados separadamente, pois, caso contrário, a cadeia ·123+4, por exemplo, passaria por um número decimal!

Se o número é representado em notação científica ou em qualquer outra que inclua uma forma exponencial do número, é necessário conjugar este teste com outros, pois um número nessa forma seria rejeitado pelo presente teste.

```
      INPUT "Introduza o no. como u
       ma cadeia:";A$

      LET t=1: IF A$(1)="+" OR A$(1
       )="-" THEN LET t=t+1
      FOR p=t TO LEN A$: IF A$(p)>=
       "0" AND A$(p)<="9" OR A$(p)=
       "." THEN GO TO A
      PRINT A$;" nao e'um numero de
       cimal": STOP
A     NEXT p
B     RETURN
B     PRINT "OK"
```

54 TESTE PARA UM NÚMERO BINÁRIO

Esta sub-rotina testa um número binário inteiro, positivo ou negativo, e pode ser utilizada, por exemplo, antes de uma rotina de conversão. É semelhante à sub-rotina de teste de um número decimal, embora mais restrita, pois apenas admite números inteiros.

Tal como na sub-rotina anterior, a segunda linha (IF CODE A$=ø...) serve para evitar que o programa «estoire» se a tecla ENTER for pressionada acidentalmente antes de o número ter sido introduzido.

```
      INPUT "Introduza o no. como u
       ma cadeia:";A$

       LET t=1: IF A$(1)="+" OR A$(
       1)="-" THEN LET t=t+1
      FOR p=t TO LEN A$
      IF A$(p)<>"0" AND A$(p)<>"1"
      THEN GO TO A
      NEXT p: GO TO R
A     PRINT A$;" nao e'um no. binar
       io": STOP
R     RETURN

R     PRINT "OK"
```

55 SUBLINHAR

O traço para sublinhar disponível no teclado do *Spectrum* (tecla Ø) é formado pela última fila de *pixels* de uma célula de carácter e, por isso, fica pegada à parte inferior de uma letra que seja formada nessa célula de carácter (cada letra ocupa geralmente um quadrado de 6×6 *pixels* no centro de cada célula de 8×8 *pixels*, ficando assim uma margem de um *pixel* em torno da letra).

Assim, estas sub-rotinas utilizam a primeira fila de *pixels* do quadrado (célula de carácter) imediatamente inferior para desenhar o traço que deve sublinhar uma dada letra, deixando assim um espaço visível (um *pixel*) entre eles. A explicação para as declarações POKE utilizadas encontra-se nas notas referentes ao «Ficheiro de imagem» (18). A declaração PRINT que se encontra na penúltima linha de cada sub-rotina destina-se a deixar um espaço em branco sob o cabeçalho que acaba de ser impresso e sublinhado.

Sub-rotina 55.1 — Imprime um cabeçalho centrado no *écran* e sublinha-o com uma ou duas linhas contínuas.

```
    INPUT "Cabecalho a sublinhar?
      ";A$;"Linha no.? ";H: REM H
      <7
    INPUT "Uma ou duas linhas? ";
     p


    PRINT AT H,16-LEN A$/2;A$
    FOR n=1 TO LEN A$
    POKE 16431+32*H+n-LEN A$/2,25
      5
    IF p=2 THEN POKE 16943+32*H+n
     -LEN A$/2,255
    NEXT n: PRINT
R   RETURN
```

Exemplo

SUBLINHAR

SUBLINHAR

SUBLINHAR

Nota: Substituir 255 por 6Ø para se obter uma linha tracejada.

Sub-rotina 55.2 — Imprime um cabeçalho centrado no *écran* e sublinha-o com uma linha que apresenta uma descontinuidade sob qualquer espaço em branco existente no cabeçalho.

```
      INPUT "Cabecalho a sublinhar?
       ";A$,"Linha no.? ";H: REM H<
       7

      PRINT AT H,16-LEN A$/2;A$
       FOR n=1 TO LEN A$
      IF A$(n)<>" " THEN POKE 16431
       +32*H+n-LEN A$/2,255
      NEXT n: PRINT
R     RETURN
```

Exemplo

SUBLINHAR ESTE CABECALHO

56 ORDENAÇÃO DE PALAVRAS

Esta sub-rotina coloca uma série de palavras (as quais podem conter letras maiúsculas ou minúsculas, traços de união, pontos e apóstrofos) em ordem alfabética.

Uma rotina de ordenação, funcionando pelo método dito de «bolhas» (*bubblesort* — ver rotina 5), ordena as palavras pela ordem do código (CODE) ASCII dos caracteres que as constituem. Contudo, para se conseguir uma ordem alfabética correcta, é necessário reduzir todas as maiúsculas a minúsculas ou vice-versa. Outros símbolos, que não letras, incluídos nas palavras, não têm relevância para a ordenação alfabética e devem ser por ela ignorados.

O quadro de cadeias Z$ é utilizado para armazenar as palavras originais, ou seja, na sua versão original. B$ destina-se às «versões modificadas», contendo apenas letras minúsculas e maiúsculas «reduzidas» a minúsculas, mas excluindo todos os caracteres que não sejam letras. Este quadro B$ é ordenado pela linha E, arrastando o quadro Z$ na ordenação. Deste modo, a ordenação propriamente dita é feita no quadro que contém as palavras constituídas apenas por letras minúsculas e desprovidas de quaisquer outros caracteres, comandando esta operação a ordenação correspondente de Z$.

```
INPUT "Numero de palavras? ";
 n

DIM B$(n+1,15): DIM Z$(n+1,15
 )
FOR q=1 TO n: INPUT "A palavr
 a ";(q);" e': ";A$;: LET Z$(
 q)=A$
FOR p=1 TO LEN A$: LET t=1
IF A$(p)="-" OR A$(p)="" OR A
 $(p)="," OR A$(p)="." OR A$(
 p)="'" OR A$(p)>"A" AND A$(
 p)<"Z" OR A$(p)>"a" AND A$
```

```
     (p)<="z" THEN NEXT p: GO TO
      A
     PRINT A$;" nao e' uma palavra
     ": LET q=q-1: GO TO B
A    FOR p=1 TO LEN A$
     IF A$(p)>="A" AND A$(p)<="Z"
      THEN LET B$(q)(t)=CHR$ (CODE
       A$(p)+32): LET t=t+1: GO TO
       C
     IF A$(p)>="a" AND A$(p)<="z"
      THEN LET B$(q)(t)=A$(p): LE
      T t=t+1
C    NEXT p
B    NEXT q
D    LET q=0: FOR p=1 TO n-1
     IF B$(p+1)<B$(p) THEN GO SUB
      E: LET q=q+1
     NEXT p: IF q<>0 THEN GO TO D
     GO TO R
E    LET C$=B$(p): LET B$(p)=B$(p+
      1): LET B$(p+1)=C$
     LET C$=Z$(p): LET Z$(p)=Z$(p+
      1): LET z$(p+1)=C$
     RETURN
R    RETURN


R    FOR p=1 TO n: PRINT Z$(p): NE
      XT p
```

Exemplos

```
Numero de palavras? 6


Cinquenta sub-rotinas para o ZX
Spectrum



Cinquenta
o
para
Spectrum
sub-rotinas
ZX
```



```
Numero de palavras? 36

 A maquina nao tem qualquer pre-
tensao de criar. Pode fazer tudo
o que se lhe pedir, mas nunca
tera' o poder de antecipar uma
relacao. A sua unica competencia
e' ajudar-nos a encontrar.

                    Ada of Lovelace


A
A
a
Ada
ajudar-nos
antecipar
competencia
criar
de
de
e'
encontrar
fazer
lhe
Lovelace
maquina
mas
nao
nunca
o
o
of
pedir
Pode
poder
pretensao
qualquer
que
relacao
se
sua
tem
tera'
tudo
uma
unica
```

57 CRISTAIS CÚBICOS

As faces de uma estrutura cristalina, ou planos cristalográficos, são identificadas por três números inteiros, h, k e l. Estes números são os inversos das intersecções de uma face com os eixos x, y e z de um sistema de coordenadas ortogonais (no caso de cristais cúbicos), e designam-se por índices de Miller do plano ou face considerada, sendo usada a notação (h k l) para indicar a face. Um plano paralelo a um eixo intersecta esse eixo no infinito sendo, portanto, nulo o índice respectivo. Uma intersecção negativa é, por sua vez, indicada por um traço colocado sobre o índice correspondente, como, por exemplo, em (h k l).

O símbolo (h k l) designa todo o conjunto de planos (ou faces) paralelos h k l da estrutura cristalina, enquanto os planos dessa mesma estrutura (ou tipo cristalográfico) com orientações diferentes são designados por {h k l}. Assim, por exemplo, {1 1 1} designa todos os planos octaédricos da rede cúbica, ou sejam, os planos (1 1 1), (1 1 $\bar{1}$), (1 $\bar{1}$ 1), ($\bar{1}$ 1 1), ($\bar{1}$ $\bar{1}$ $\bar{1}$), ($\bar{1}$ $\bar{1}$ 1), ($\bar{1}$ 1 $\bar{1}$), e (1 $\bar{1}$ $\bar{1}$), formando um octaedro (figura), enquanto {1 $\emptyset$ $\emptyset$} designa os planos ($\bar{1}$ $\emptyset$ $\emptyset$), ($\emptyset$ 1 $\emptyset$), ($\emptyset$ $\bar{1}$ $\emptyset$), ($\emptyset$ $\emptyset$ 1) e ($\emptyset$ $\emptyset$ $\bar{1}$), os quais formam um cubo. Do mesmo modo, {1 1 $\emptyset$} designa os doze planos que formam um dodecaedro rômbico, etc.

Este programa determina, em primeiro lugar, um quadro de controle, para permitir que as linhas correctas sejam desenhadas e, depois, solicita-nos os valores de h, k e l que queiramos seleccionar. Seguidamente, calcula as coordenadas dos vértices da figura e une esses vértices numa projecção 3-D por meio de linhas a cheio (parte da frente da figura) e linhas tracejadas (parte de trás da figura).

A figura é então designada por h k l, impressa no canto superior esquerdo do *écran*, copiada pela impressora, se quisermos, e o programa solicita os próximos valores para h, k e l.

Na ausência de *microdrive*, a linha 5 gera uma linha $\emptyset$.

As linhas 1$\emptyset\emptyset$ a 14$\emptyset$ constroem uma matriz 3×6×8 com todas as permutações de h, k e l, com valores positivos e negativos, os quais são armazenados temporariamente em 1(3,8,6).

As linhas 15Ø a 18Ø seleccionam 26 desses ternos ordenados, os quais irão ser necessários posteriormente, ficando estes armazenados em A(26,3,3).

Para determinar as coordenadas de um vértice, o programa escolhe três faces que se intersectem nesse vértice, e as suas equações respectivas são resolvidas num sistema de equações. A solução desse sistema é constituída pelas coordenadas do vértice em questão, pois ele é o único ponto comum aos três planos.

Isto efectua-se na linha 21Ø, onde são seleccionados, à vez, os 26 grupos de três planos. A linha 22Ø calcula o determinante da matriz assim formada, o qual deve ser diferente de zero, e a linha 23Ø resolve o sistema de equações. A solução é armazenada em 1(26,3).

A linha 25Ø utiliza a fórmula da projecção (sub-rotina 36) para obter A(26,2) e as coordenadas X e Y de cada vértice no plano do *écran,* sendo h utilizado como factor de escala.

As linhas 4ØØ a 49Ø separam as faces em sete tipos distintos, nomeadamente:

h k l
h l l
h h l
h h h
h k Ø
h h Ø
h Ø Ø

As sub-rotinas 500, 600, 700, 800 e 900 decidem quais os pontos a unir com linhas a cheio ou tracejadas. Num cubo, por exemplo, só os cantos adjacentes devem ser unidos por linhas, e não devem ser traçadas diagonais de faces ou internas. A sub-rotina 1500 destina-se a traçar uma linha a cheio e a sub-rotina 1600 uma linha tracejada. Estas decisões são tomadas com base nos valores do quadro de controle C(26,3), o qual é construído no início da execução do programa, por meio da inserção de (1 1 1), (1 1 Ø) e (1 Ø Ø), efectuada pelos cálculos mencionados acima, em três ciclos controlados por g. Quando C(26,3) é completado, o programa solicita os valores para h, k e l, por meio da linha 4Ø, e procede seguidamente à sua ordenação decrescente nas linhas 5Ø e 6Ø, antes de calcular e desenhar a forma cristalina {h k l}.

Como auxiliares ao estudo e introdução do programa são apresentados em anexo o fluxograma principal e o fluxograma relativo à separação das faces cristalinas, além do quadro de controle C(26,3) e a matriz A(26,3,3) dos ternos ordenados indicativos das faces.

De referir que o programa apresenta uma execução um tanto ou quanto lenta, levando cerca de 65 segundos para estabelecer o quadro de controle no início e 32 segundos para calcular cada forma cristalina. Tal como é apresentado, o programa ocupa cerca de 4,2 k *bytes* de memória.

```
  0>REM "CRISTAIS CUBICOS"
  5 POKE 23756,0: REM © W.J.
 10 CLEAR 32767: LET j=1: LET f
=2: LET e=3: LET q=6: LET m=8: L
ET o=26: LET g=1200: LET h=j: LE
T k=j: LET l=j: LET cont=0: DIM
C(o,e): PRINT FLASH 1;"AGUARDE":
 GO TO 100
 20 IF g=1226 THEN LET l=0: GO
TO 100
 30 IF g=1252 THEN LET k=0: GO
TO 100
 40 PRINT AT 0,0;" {";h;k;l;"}
": IF cont=1 THEN GO SUB 1750
 50 INPUT "h=";h;" k=";k;" l=";
l: LET cont=1
 60 CLS : IF k>h THEN LET a=h:
LET h=k: LET k=a
 70 IF l>k THEN LET a=k: LET k=
l: LET l=a: GO TO 60
 80 PRINT FLASH 1;"Calculos a s
erem efectuados"
 100 DIM l(e,m,q): DIM A(m,e): F
OR a=j TO 4: LET A(a,j)=j: LET A
(a+4,j)=-j: NEXT a
 110 RESTORE 110: FOR a=j TO m:
READ A(a,f): NEXT a: DATA j,-j,-
j,j,-j,j,j,-j
 120 FOR a=j TO q STEP 4: LET A(
```

```
a,e)=j: LET A(a+j,e)=j: LET A(a+
f,e)=-j: LET A(a+e,e)=-j: NEXT a
 130 DIM D(18): RESTORE 130: FOR
 a=j TO 18: READ D(a): NEXT a: D
ATA h,k,l,k,l,h,l,h,k,l,k,h,h,l,
k,k,h,l
 140 FOR b=j TO m: LET n=j: FOR
 c=j TO q: FOR a=j TO e: LET l(a,
b,c)=A(b,a)*D(n): LET n=n+j: NEX
T a: NEXT c: NEXT b
 150 DIM D(j): DIM A(o,e,e): FOR
 c=j TO e: FOR a=j TO e: LET A(c
,j,a)=l(a,j,(c=j)+(c=f)*f+(c=e)*
q): LET A(c,f,a)=l(a,(c=j)*f+(c=
f)*q+(c=e)*q,(c=j)+(c=f)*f+(c=e)
*q): LET A(c,e,a)=l(a,f*c+j,q-c)
: NEXT a: NEXT c
 160 FOR c=4 TO q: FOR a=j TO e:
 LET A(c,j,a)=l(a,(7-c)*f,(c=4)+
(c=5)*f+(c=q)*q): LET A(c,f,a)=l
(a,(c=4)*5+(c=5)*7+(c=q)*5,(c=4)
+(c=5)*f+(c=q)*q): LET A(c,e,a)=
l(a,m,9-c): NEXT a: NEXT c
 170 FOR b=7 TO 14: FOR c=j TO e
: FOR a=j TO e: LET A(b,c,a)=l(a
,b-q,c): NEXT a: NEXT c: NEXT b
 180 FOR c=15 TO o: FOR a=j TO e
: LET A(c,j,a)=l(a,INT (c/e-5)*f
+j,(c/e-INT (c/e))*e+j): LET A(c
,f,a)=l(a,INT (c/e-5)*f+j,-((c/e
-INT (c/e))*e-q)): LET A(c,e,a)=
l(a,((c=16) OR (c=18) OR (c=23))
*f+((c=15) OR (c=19) OR (c=o))*4
+((c=17) OR (c=22) OR (c=24))*q+
((c=20) OR (c=21) OR (c=25))*m,(
c/e-INT (c/e))*e+j): NEXT a: NEX
T c
 200 IF g<1278 THEN GO SUB g: LE
T g=g+o: GO TO 20
 210 DIM l(o,e): FOR a=j TO o: L
ET r=A(a,j,j): LET s=A(a,j,f): L
ET t=A(a,j,e): LET u=A(a,f,j): L
ET v=A(a,f,f): LET w=A(a,f,e): L
ET x=A(a,e,j): LET y=A(a,e,f): L
ET z=A(a,e,e)
 220 LET D=r*(v*z-y*w)+s*(w*x-z*
v)+t*(u*y-x*v): IF D=0 THEN GO T
O 240
 230 LET l(a,j)=(v*z-y*w+s*(w-z)
+t*(y-v))/D: LET l(a,f)=(r*(z-w)
+w*x-u*z+t*(u-x))/D: LET l(a,e)=
(r*(v-y)+s*(x-u)+(u*y-x*v))/D
 240 NEXT a
```



```
 250 DIM A(o,f): FOR a=j TO o: L
ET A(a,j)=71.95*h*l(a,f)-24*h*l(
a,j)+128: LET A(a,f)=72*h*l(a,e)
-8*h*l(a,j)-2.672*h*l(a,f)+84: N
EXT a
 400 DIM l(j): CLS : IF h=k THEN
 GO TO 460
 410 IF l<>0 THEN GO TO 440
 420 GO SUB 700: IF k<>l THEN GO
 SUB 800
 430 GO TO 40
 440 GO SUB 500: GO SUB 600: IF
k<>l THEN GO SUB 800
 450 GO TO 40
 460 IF l<>0 THEN GO SUB 900: GO
 TO 480
 470 GO SUB 800: GO TO 40
 480 IF l<>k THEN GO SUB 800
 490 GO TO 40
 500 FOR a=j TO q: FOR b=j TO e:
 IF c(a,b)<>0 THEN GO TO 520
 510 NEXT b
 520 FOR c=15 TO o: IF c((c,b)<>c
(a,b) THEN GO TO 550
 530 IF c((c,j)=-j OR a=4 THEN GO
 SUB 1600: GO TO 550
 540 GO SUB 1500
 550 NEXT c: NEXT a: RETURN
 600 FOR a=7 TO 14: FOR c=15 TO
o: IF (c((c,j)=c(a,j) OR c((c,j)=0
) AND (c((c,f)=c(a,f) OR c((c,f)=0
) AND (c((c,e)=c(a,e) OR c((c,e)=0
) THEN GO TO 620
 610 GO TO 640
 620 IF A>10 AND c>=20 THEN GO S
UB 1600: GO TO 640
 630 GO SUB 1500
 640 NEXT c: NEXT a: RETURN
 700 FOR a=7 TO 14 STEP 7: FOR c
=7 TO 14: IF c(a,j)=-c((c,j) AND
c(a,f)=c((c,f) AND c(a,e)=c((c,e)
OR c(a,j)=c((c,j) AND c(a,f)=-c((c
,f) AND c(a,e)=c((c,e) OR c(a,j)=
c((c,j) AND c(a,f)=c((c,f) AND c(a
,e)=-c((c,e) THEN GO TO 720
 710 GO TO 740
 720 IF c=14 THEN GO SUB 1600: G
O TO 740
 730 GO SUB 1500
 740 NEXT c: NEXT a: RETURN
 800 FOR a=7 TO 14: FOR b=j TO e
: FOR c=j TO q: IF c(a,b)=c((c,b)
 THEN GO TO 820
```

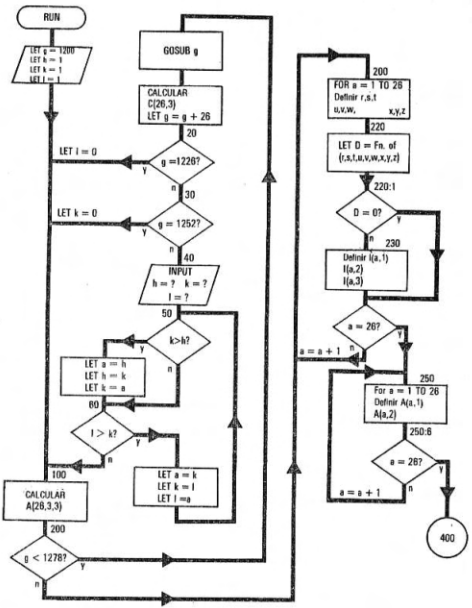
```
 810 NEXT c
 820 IF a>10 THEN GO TO 840
 830 GO SUB 1500: GO TO 860
 840 IF c(a,f)=-j OR c(c,j)=-j T
HEN GO SUB 1600: GO TO 860
 850 GO TO 830
 860 NEXT b: NEXT a: RETURN
 900 FOR a=j TO q: FOR b=j TO e:
 IF c(a,b)<>0 THEN GO TO 920
 910 NEXT b
 920 FOR c=j TO q: IF b=e THEN G
O TO 950
 930 IF c(c,b+j)<>0 THEN GO TO 9
70
 940 GO TO 990
 950 IF c(c,j)<>0 THEN GO TO 970
 960 GO TO 990
 970 IF c=4 OR a=4 THEN GO SUB 1
600: GO TO 990
 980 GO SUB 1500
 990 NEXT c: NEXT a: RETURN
1200 FOR a=7 TO 14: GO SUB 1300:
 NEXT a: RETURN
1225 FOR a=15 TO 0: GO SUB 1300:
 NEXT a: RETURN
1252 FOR a=j TO q: GO SUB 1300:
NEXT a: RETURN
1300 FOR b=j TO e: LET c(a,b)=A(
a,j,b): NEXT b: RETURN
1500 PLOT A(a,j),A(a,f): DRAW A(
c,j)-A(a,j),A(c,f)-A(a,f): RETUR
N
1600 IF A(c,j)=A(a,j) THEN GO TO
 1680
1610 IF A(c,f)=A(a,f) THEN GO TO
 1650
1620 LET p=(A(c,f)-A(a,f))/(A(c,
j)-A(a,j))
1630 IF ABS p<j THEN GO TO 1660
1640 IF ABS p>=j THEN LET p=j/p:
 GO TO 1690
1650 LET p=0
1660 LET i=10*SGN (A(c,j)-A(a,j)
)/SQR (j+(ABS p)↑f)
1670 FOR n=j TO 1+(A(c,j)-A(a,j)
)/i: PLOT A(a,j)+(n-j)*i,A(a,f)+
(n-j)*i*p: DRAW i*.4,i*.4*p: NEX
T n: RETURN
1680 LET p=0
1690 LET i=10*SGN (A(c,f)-A(a,f)
)/SQR (j+(ABS p)↑f)
1700 FOR n=j TO 1+(A(c,f)-A(a,f)
)/i: PLOT A(a,j)+(n-j)*i*p,A(a,f
```



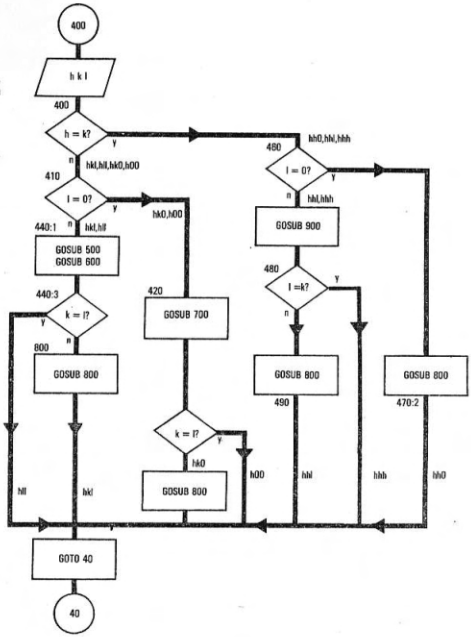
```
)+(n-j)*i: DRAW i*.4*p,i*.4: NEX
T n: RETURN
1750 PRINT #0;"Imprimir = I
     Terminar = T"
1760 IF INKEY$="" THEN GO TO 176
0
1770 IF INKEY$="i" OR INKEY$="I"
 THEN COPY
1780 IF INKEY$="t" OR INKEY$="T"
 THEN STOP
1790 RETURN
```

Quadros de controle dos cristais cúbicos e matriz de ternos ordenados das faces.

n	C(26,3)		A(26,3,3)	
1	1 0 0	h k l	h -k l	h -l -k
2	0 0 1	k l h	-k l h	-l -k h
3	0 1 0	k h l	-k h l	-l h -k
4	-1 0 0	-h k l	-h -k l	-h -l -k
5	0 0 -1	k l -h	-k l -h	-l -k -h
6	0 -1 0	k -h l	-k -h l	-l -h -k
7	1 1 1	h k l	k l h	l h k
8	1 -1 1	h -k l	k -l h	l -h k
9	1 -1 -1	h -k -l	k -l -h	l -h -k
10	1 1 -1	h k -l	k l -h	l h -k
11	-1 -1 1	-h -k l	-k -l h	-l -h k
12	-1 1 1	-h k l	-k l h	-l h k
13	-1 1 -1	-h k -l	-k l -h	-l h -k
14	-1 -1 -1	-h -k -l	-k -l -h	-l -h -k
15	1 1 0	h k l	k h l	h k -l
16	1 0 1	k l h	h l k	k -l h
17	0 1 1	l h k	l k h	-l h k
18	1 -1 0	h -k -l	k -h -l	h -k l
19	1 0 -1	k -l -h	h -l -k	k l -h
20	0 -1 -1	l -h -k	l -k -h	-l -h -k
21	-1 -1 0	-h -k l	-k -h l	-h -k -l
22	-1 0 1	-k -l h	-h -l k	-k l h
23	0 -1 1	-l -h k	-l -k h	-l h k
24	-1 1 0	-h k -l	-k h -l	-h k l
25	-1 0 -1	-k l -h	-h l -k	-k -l -h
26	0 1 -1	-l h -k	-l k -h	l h -k

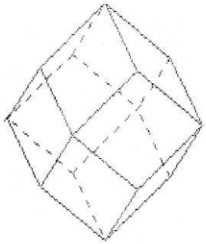


Fluxograma do programa dos cristais cúbicos



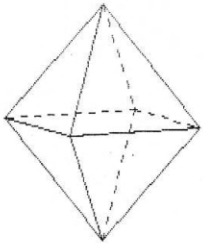
Fluxograma da separação das faces cristalinas

{110}

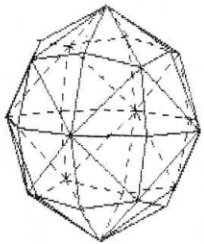


Formas cristalinas

{111}



{421}



Forma cristalina

Biblioteca Verbo de Informática

1.° volume

Jogos Dinâmicos para o ZX Spectrum
de *Tim Hartnell*

Simultaneamente com programas de diversão e de criatividade, esta obra oferece a possibilidade de um perfeito conhecimento e do uso consciente do computador. Inclui jogos de «arcada» e de tabuleiro e programas de aventuras e sugestões para melhorar e desenvolver a programação.

2.° volume

Aprofundar o Basic do Spectrum
de *Mike Lord*

Simultaneamente obra de carácter didáctico muito sério e meio de diversão para o amador de informática, esta obra interessa a um largo leque de utilizadores do *Spectrum*, desde principiantes a programadores experientes.
É fonte de referência imprescindível quanto ao *software* desta máquina.

3.° volume

O Domínio do Código Máquina do Spectrum
de *Toni Baker*

Este livro satisfaz a ambição – e a necessidade – de todos os programadores que utilizam o *Spectrum:* dominar directamente a linguagem que o coração da máquina conhece, o que representa um salto qualitativo quanto a versatilidade de programação, velocidade de execução e domínio do grafismo.

4.° volume

As Melhores Rotinas para o ZX Spectrum
de *John Hardman e Andrew Hewson*

40 rotinas em código máquina para o *Spectrum*. Obra destinada a quem quer tirar o melhor partido do seu minicomputador: utilizando a linguagem máquina do *Spectrum*, multiplicam-se espantosamente as suas capacidades.
Além disso, trata-se de um instrumento de trabalho de inexcedível utilidade.

5.º volume

Os 20 Melhores Pogramas para o Spectrum
de *Andrew Hewson*

Estes programas, além da sua utilidade prática, representam uma perfeita fonte de referência das técnicas de programação mais difundidas e ensinam a arte de programar através de exemplos perfeitos.

6.º volume

Guia Avançado para o Spectrum
de *Mike James*

Introdução prática às características mais avançadas do *Spectrum*, tanto no *hardware* como no *software*.
Oferece ao leitor a exploração das possibilidades mais sofisticadas deste microcomputador.